|  |  |
| --- | --- |
| **Grade:** | X / 100 |
| **Instructor’s Comments:** | To be filled by the instructor |

This lab template is developed per data collection requirement for ABET accreditation process. Your cooperation of using it is highly appreciated. ***Please do not delete this page!***

**Grading Criteria**:

Each lab is worth 100 points. The first 20 points are toward your submission of required documents, and the rest 80 points are for the problem of each lab assignment.

|  |  |  |
| --- | --- | --- |
| **Item** | **Description** | **Points** |
| Required documents | 1) The lab report (this document)  **2) Any source code files (Don’t forget to compress them into a “.zip” file!)** | 20 |
| For all exercises/problems | Completeness of your work  1) Complete solution  2) Inclusion of screenshot of executing your code – you must include at least one whether your code works or not (if applicable)  Correctness of your logic/solution  Coding style (below are common mistakes)  1) Proper alignment of your code  2) Proper naming convention  3) Meaningful naming | 80 |

**ITS 330 – Advanced Operating Systems**

**Lab 08**

**Due: April 8th, 2021**

**Laquon Hamilton**

(***Create a table of contents before your solution***)

Contents

[Problem 1. 3](#_Toc65172166)

[Problem 2. 9](#_Toc65172167)

[Problem 3. 9](#_Toc65172168)

[Problem 4. 10](#_Toc65172169)

[Problem 5. 10](#_Toc65172170)

[Problem 6. 10](#_Toc65172171)

[Problem 7. 10](#_Toc65172172)

[Problem 8. 10](#_Toc65172173)

[Problem 9. 11](#_Toc65172174)

# Problem 1.

1. What is the value of balance when you run it?

- It alternates between 6 and 7

1. Do you think your output is correct? If not, explain why the data is corrupt.

- No, each thread is trying to access the deposit method at the same time

1. Please revise the code and guarantee the balance is always 100 after all threads are finished no matter how many times the code is executed.

Code:

Part 1

/\*

 ITS-330

 Lab 08

 Problem 1

 4/7/2021

 Laquon Hamilton

\*/

import java.util.concurrent.\*;

public class AccountWithSync

{

private static Account account = new Account();

public synchronized static void main(String[] args) {

ExecutorService executor = Executors.newCachedThreadPool(); //create thread pool

// Create and launch 100 threads

for (int i = 0; i < 100; i++) {

executor.execute(new AddAPennyTask());

}

executor.shutdown();

// Wait until all tasks are finished

while (!executor.isTerminated()) {

}

System.out.println("What is balance? " + account.getBalance());

}

// A thread for adding a penny to the account

private static class AddAPennyTask implements Runnable, java.lang.Runnable {

public synchronized void run() {

account.deposit(1);

}

}

// An inner class for account

private static class Account {

private int balance = 0;

public int getBalance() {

return balance;

}

public synchronized void deposit(int amount) {

int newBalance = balance + amount;

// This delay is deliberately added to magnify the

// data-corruption problem and make it easy to see.

try {

Thread.sleep(5);

}

catch (InterruptedException ex) {

}

balance = newBalance;

System.out.println("Thread " + balance + ":" + balance);

}

}

public interface Runnable

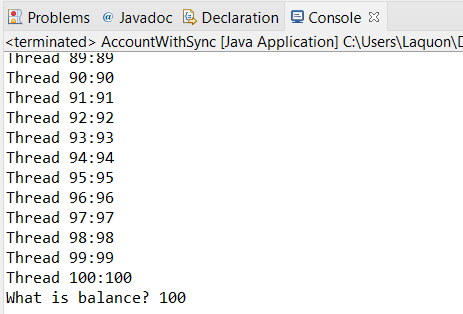
{

  public abstract void run();

}

}

Screenshot



# Problem 2.

Code

/\*

 ITS-330

 Lab 08

 Problem 2

 4/7/2021

 Laquon Hamilton

\*/

import java.util.concurrent.\*;

public class Increment1000

{

private static Increment incremental = new Increment();

public synchronized static void main(String[] args) {

ExecutorService executor = Executors.newCachedThreadPool(); //create thread pool

// Create and launch 1000 threads

for (int i = 0; i < 1000; i++) {

executor.execute(new IncrementTask());

}

executor.shutdown();

// Wait until all tasks are finished

while (!executor.isTerminated()) {

}

System.out.println("Total value: " + incremental.getValue());

}

// A thread for adding a penny to the account

private static class IncrementTask implements Runnable, java.lang.Runnable {

public synchronized void run() {

incremental.addValue(1);

}

}

// An inner class for account

private static class Increment {

private int totalValue = 0;

public int getValue() {

return totalValue;

}

public synchronized void addValue(int amount) {

int newBalance = totalValue + amount;

// This delay is deliberately added to magnify the

// data-corruption problem and make it easy to see.

try {

Thread.sleep(5);

}

catch (InterruptedException ex) {

}

totalValue = newBalance;

System.out.println("Thread " + totalValue + ":" + totalValue);

}

}

public interface Runnable

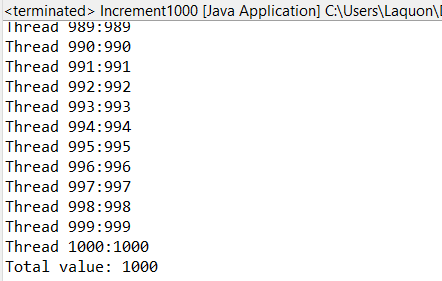
{

  public abstract void run();

}

}

Screenshot



# Problem 3.

Code:

Screenshot

# Problem 4.

Code

Screenshot

# Problem 5.

Code

Screenshot

# Problem 6.

Code

Screenshot

# Problem 7.

Code

Screenshot

# Problem 8.

Code

Screenshot

# Problem 9.

Code

Screenshot