Parallel and MultiThreaded Programming

CSYE 7215

Homework 6

Due: October 24, 2020

Put all your java, compiled class files and documentation into zip file named Homework6.zip and submit it via the dropbox on the canvas before the END of due date. Put your name on all .java files. There will be a short quiz on this assignment.

1.Provide Descriptions:Exx

**Parallelism versus Concurrency, provide diagram**

Parallelism :

Parallelism means that an **application splits its tasks up into smaller subtasks which can be processed in parallel**, for **instance on multiple CPUs at the exact same time.**

Task 1

Sub task 1

Sub task 1

Sub task 1

Concurrency :

Concurrency means that an **application is making progress on more than one task at the same time (concurrently).** if the computer only has one CPU the application may not make progress on more than one task at exactly the same time, but more than one task is being processed at a time inside the application. It does not completely finish one task before it begins the next. Instead, the CPU switches between the different tasks until the tasks are complete.

Task 2

Task 3

Task 1

All task is running concurrently.

**CPU core, How do you make cpu core?**

A CPU core is the part of something central to its existence or character. In the same way in the computer system, the CPU is also referred to as the core.

There are basically two types of core processor:

* Single-Core Processor
* Multi-Core Processor

CPU Core recieve the instruction and perform the operation

A core can work on one task, while another core works a different task, so the more cores a CPU has, the more efficient it is.

In parallel execution, the tasks to be performed by a process are broken down into sub-parts, and multiple CPUs (or multiple cores) process each sub-task at precisely the same time.

**Adding more core means designing more chip in CPU**

**32-bit versus 64-bit architecture, give example**

* A 32-bit processor includes a 32-bit register, which can store 2^32 or 4,294,967,296 values.
* A 64-bit processor includes a 64-bit register, which can store 264 or 18,446,744,073,709,551,616 values. Therefore, a 64-bit register is not twice as large as a 32-bit register, but is 4,294,967,296 times larger.
* The CPU register stores memory addresses, which is how the processor accesses data from RAM. One bit in the register can reference an individual byte in memory, so a 32-bit system can address a maximum of 4 gigabytes (4,294,967,296 bytes) of RAM. The actual limit is often less – around 3.5 gigabytes – since part of the registry is used to store other temporary values besides memory addresses.
* A 64-bit register can theoretically reference 18,446,744,073,709,551,616 bytes, or 17,179,869,184 gigabytes (16 exabytes) of memory.
* This is several million times more than an average workstation would need to access. What's important is that a 64-bit computer (which means it has a 64-bit processor) can access more than 4 GB of RAM. **If a computer has 16 GB of RAM, it better have a 64-bit processor. Otherwise, at least 12 GB of the memory will be inaccessible by the CPU.**
* While 64 bits is far more storage than what modern computers require, it removes all bottlenecks associated with 32-bit systems. For example, 64-bit systems run more efficiently since memory blocks are more easily allocated. They also support 64-bit instructions and have 64-bit data paths, which enables them to process more data at once than 32-bit systems can.

**Client/Server Socket programming**

* Client and server programs need to be able to establish connections to each other over the network. Client and server applications connect through methods defined by Transmission Control Protocol/Internet Protocol -- or TCP/IP -- standards.
* The server is linked or bound to a specific IP address on the network that is reachable by the client.
* Once a connection is established, the client sends the server whatever data and instructions are needed to complete the tasks it needs the server to perform.
* The client application takes the data it receives from the server, displays it, saves it to a storage device, prints it or forwards it to other clients on the network.
* The server program plays a passive role: it constantly checks its network connection to see if there are any clients sending it requests for a service it can provide.
* Once the server program detects a service request, it establishes the socket and begins receiving and sending information to the client until the job is done.
* Finally, the server closes the connection to the client and resumes the wait for more client program requests for services.

2. You can run tasks in parallel using Java8 Collection.parallelStream. Consider ParallelExample2, ParallelExample3a, ParallelExample3b, and ParallelExample5. Compile and run. Explain how the code works in each program and discuss the Results.

ParallelExample2:

* In This Example we are printing the value of a to z using stream() and parallel stream.
* Output of stream() method is **in sequence** on the other hand output of parallel stream is **not in sequential manner.**

Output

A screenshot of a computer

Description automatically generated

**ParallelExample3a**

🡪In This example first we are generating range of values .

🡪 First we are printing the values with For Each and verifying that operation is parallel or not using isParallel() method.

🡪 In Output we can clearly check that values are printing in sequential manner

🡪 after that we are making that range parallel using parallel() method and we can observe the output is not printing the value in sequential manner.

A close up of a logo

Description automatically generated

**ParallelExample3b**

Text

Description automatically generated

🡪 In Example 3b are printing the currently running thread name and we can see that the first foreach loop is ran by Main- thread only while parallel method creates the multiple threads using forkJoin threadpool and print the values.

**ParallelExample5**

🡪 In this example we are comparing the performance of the sequential vs parallel method

* We have created one employee class with all the employee details.
* Inside the main method we are creating **10000** employee and store the data of each employee in different file
* First time we are creating 10000 employee’s files using **stream() (sequential) method and observer the running time.**
* **In second time we will run the program using parallel stream and observer running time.**
* **If compare the running time of both the program we can clearly say that parallel stream is taking ¼ time of stream method.**

A screenshot of a computer screen

Description automatically generated

3. Write Java parallelism code to sort each array entry in two dimensional

array input data. Note: You need to use Java8 Collection.parallelStream, and run

eight threads in parallel.

int[][] arr = { { 9, 12, 6, 14, 10, 21, 13}, { 3, 5, 41, 16, 14, 10, 21},

{ 3, 15, 41, 17, 11, 10, 51}, { 3, 15, 41, 17, 11, 10, 51},

{ 4, 15, 35, 17, 11, 12, 55}, { 2, 16, 31, 18, 12, 11, 42},

{ 2, 15, 35, 10, 11, 12, 19}, { 1, 20, 33, 18, 12, 13, 44} };

* To use Collection.parallel stream first we need to store all the values in list anf then we can sort the list using parallelStream.

**List.ParrallelStream().sort()**

Output:

Text

Description automatically generated

4. Consider Java AtomicInteger example in this article, initially Part-1 is implemented Without

integer atomic operation, and part-2 is implemented With integer atomic operation.

https://www.journaldev.com/1095/atomicinteger-java

a) Compile and run Part-1, report and explain the results and code Problem

* In Part-1 we have are trying to update the value of shared resource without implementing any locking mechanism.
* As we can see in output that the value of count is varying.

Graphical user interface, text

Description automatically generated

b) Fix the problem in Part-1 without Atomic Operation, Compile and run

* We can fix this problem by putting count increament operation inside synchronized block or by making getcount method synchronized.



c) Compile and run Part-2 program which uses Atomic operation

* In part 2 we have used the AtomicIntger count variable which will make sure that the increment operation is atomic by using **increamentAndGetMethod() hence every time we will get the same output.**

A screenshot of a computer screen

Description automatically generated

d) Compare results between (a) and (c), and then compare results between (b) and (c)

* In Result (a) we get the different value count variable because we have not used any locking mechanism to protect the value of count.
* In result (c) every time we will get the same value of count variable because we have used the atomic integer which make sure that the increment operation should be atomic

🡪 We will get the same result for (b) and (c) as in result be we have used the synchronized block to protect the value of count variable.

* **But result( c) is improve the performance as it has used the atomic operation hence we don’t need to worry about any blocking**

5. In socket programming, there is server side and there is client(s) side. Consider this code discussed in class: https://www.geeksforgeeks.org/introducing-threads-socket-programming-java/

a) Explain the client/server code

b) Compile the server side code and run it. Explain what you see and why?

c) Compile client-side and run it. Explain what you see and why?

**a) Explain the client/server code**

* In Given code, we are creating one server class which will handle the client accept the client request and respond to that client.
* To handle the multiple client at the same time we have created the clientHandler class which implements the runnable interface.
* So every time when new client send a request to a server server will create separate thread of client handler class which will respond to that particular client request.

**b) Compile the server side code and run it. Explain what you see and why?**

🡪 When we comple and run the sever code it will start the sever on given port number.

* Once the server started it will continuously look for any client request.
* If client try to connect server then it will accept the request and create new Thread of client hander class to process that request.

Text

Description automatically generated

**c) Compile client-side and run it. Explain what you see and why?**

* When we compile and run the client side code it will connect to sever and server will create a new thread of client handler.
* Now client handler will prompt to client that which thing you want time or date
* When client send the valid input , client handler will respond to that request.

Text

Description automatically generated

6. You have learned how to create multi-threaded Student grading system using both implicit locking and explicit locking. Now you are to create student grading system using Socket client/server programming.

A) Consider example “A Network Tic-Tac-Toe Game” in the this article, analyze,

compile and run the code: https://cs.lmu.edu/~ray/notes/javanetexamples/

Code Analysis :

🡪 we have create on TicTacToeServer class which will act as sever.In this class we are trying to connect two players.

* We have created ThreadPool to process each request in separate thread.
* Game class will validate the basic rule of the game while player class implements the runnable interface so each player have their own thread.
* While TictacToe client class has UI Part where we can load the Jframe and start the game.

Output:

A screenshot of a computer

Description automatically generated

B) Use example model to build student grading system with 40 client student threads and one server GraderThread. Each client-student thread generates 3 scores (homework, midterm, final), and each score is randomly generated between 70 to 100. The scores are stored in ScoresMap as key/value, ({homework, score}, {midterm, score}, {final, score}). Then, each student thread connects to the GraderServer (with incremental 100 milliseconds interval), and once it is connected to the server, it sends the scoresMap, studentId, firstName, lastName. Note: You need to create a thread within GraderServer to connect to each client. The GraderServer thread receives student’s information: a) parses the ScoresMap for student scores, calculates letter grade A, B, C, D, F, and stores letter grade in its local cache GradesMap (threadId, grade), and then writes it to the file “FinalGrades”, and replies grade back to the client student thread.

b) In subsequent student requests with missing/mistake grade (updated specific score in ScoresMap), the Server parses the scoresMap information, Verifies grade information in cache, calculates letter grade, updates cache, updates file, and replies the new grade back to student thread.

Note: you need to create cache key/value

Approach :

* Created GraderSever class which will act like sever and accept the request of each student.
* To handler the multiple student request concurrently I have created GradeStudentHandler class.
* Every time a student send the score to graderserver it will create the separate thread of GraderStudent handler class and calculate the letter grade and save it into grade file and local cache and at the end it will send grade it back to that particular student.
* To Handle the different types of request such as missing/mistake grade update and report new grades i will send the value of requesttype to Gradersever and it will process the request according to requesttype.
* For Implementation details please revive the code

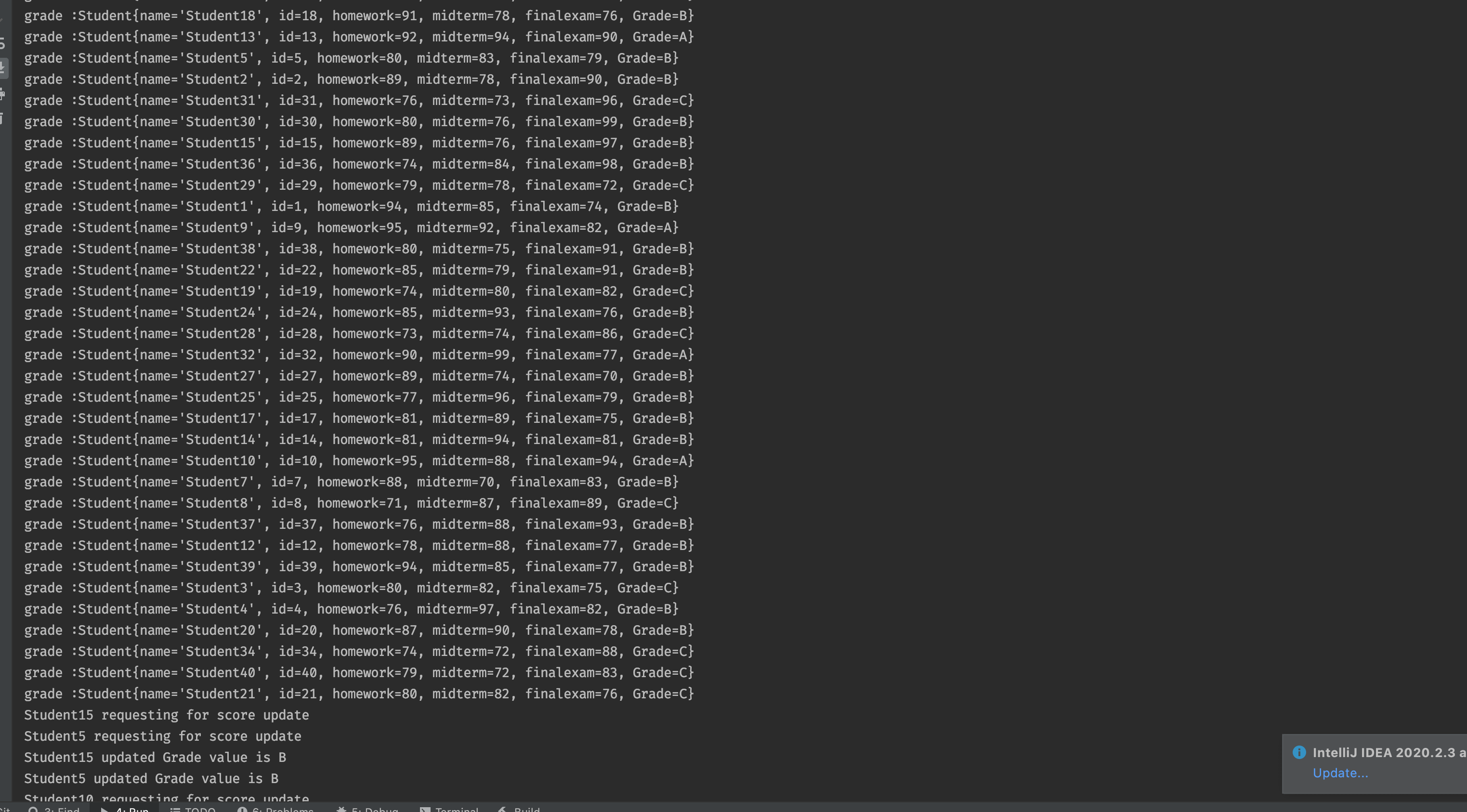
Output:

Text

Description automatically generated

A picture containing text

Description automatically generated



A screenshot of a computer

Description automatically generated

7. Consider the following ModelViewControl (MVC) model. Use this Model to identify and design three services for the University myNortheastern blackboard portal. You need to show the controller layer, service layer, and database layer for each of the services that you identify. For example: Login is a service. You provide http URL: myNortheastern/profile/login

Note: Read article: https://www.developer.com/java/data/exploring-rest-apis-with-spring-mvc.html

Diagram

Description automatically generated

**controller layer** Identify the request and then forward it to respective service layer

**Service layer** -- Connect to Database layer and manipulate data according to business logic

**Database layer** --> Established connection with database and retrieve data and pass to service layer

In Blackboard portal

**Discussion Forum**

Controller layer Service Layer

Business Logic(

All Data manipulation task according to user request

)

createNewDiscussionGroup()

PostQuestion()

PostAnswer()

DeletQuestion()

DeleteAnswer()

ListAllAnswer()

ListAnswerByQuestion()

**Data Layer contains the DB**

**Connection**

**And Model Classes such as questions , answer , post etc**

**Billing**

Controller Layer Service Layer

Add Newbill()

PayBill()

View bill()

AddCreditCard()

DeleteCreditCar()

Business Logic(

All Data manipulation task according to user request

)

**Data Layer contains the DB**

**Connection**

**And Model Classes such as studentTransactionDetail,StudentDetail,studentPaymentDetail**

**Course Registration**

Controller Layer Service Layer

AddCourse()

Dropcourse()

viewAllCourses()

viewCourseByBranchName()

Business Logic(

All Data manipulation task according to user request

)

**Data Layer contains the DB**

**Connection and model classes such as CourseDetail,**

**courseRegistrationDetail**

**BranchDetail etc..**