Final Exam

Graded

Student

AKSHAJ KAMMARI

Total Points

53.5 / 110 pts

Question 1

Streams **13** / 20 pts

1.1 (no title) **11** / 15 pts

+ 15 pts Correct

✔ **+ 1 pt** Files.lines(Paths.get("doc.txt"))

✔ **+ 2 pts** map(l -> l.split(" "))

✔ **+ 4 pts** flatMap(Arrays::stream)

+ 2 pts map(String::lowerCase())

✔ **+ 1 pt** distinct

Collect

✔ **+ 0.5 pts** collect

✔ **+ 0.5 pts** groupingBy

✔ **+ 1 pt** String::length (first param to groupingBy)

✔ **+ 1 pt** counting() (second parameter to groupingBy)

+ 2 pts Result type is Map<Integer,Long>

+ 0 pts Incorrect

Jindong Jiang

1.2 (no title) **2** / 5 pts

+ 5 pts Correct

✔ **+ 1 pt** Arrays.stream(arr)

+ 2 pts mapping to sum of columns

✔ **+ 1 pt** averaging

+ 1 pt Result type is `OptionalDouble` or `Optional<Double>`

+ 0 pts Incorrect

Question 2

Streams **26** / 30 pts

2.1 **(no title)** **7** / 11 pts

+ 11 pts Correct

✔ **+ 1 pt** artists.stream()

✔ **+ 1 pt** filter on year = 2023

Click here to replace this description.

✔ **+ 3 pts** sorted on comparing stream count

+ 1 pt reversing the default sort order

✔ **+ 1 pt** limit(10)

+ 1 pt map(Artist::getName)

+ 2 pts toArray(String[]::new)

✔ **+ 1 pt** Result type is String[]

+ 0 pts Incorrect

2.2 **(no title)** **9** / 9 pts

+ 9 pts Correct

✔ **+ 1 pt** artists.stream()

✔ **+ 1 pt** filter on streamCount >= 10000000

Collect

✔ **+ 2 pts** groupingBy first argument Artist::getGenre

✔ **+ 3 pts** groupingBy second argument mapping(Artist::getYear,toset())

✔ **+ 2 pts** Result type is `Map<Genre, Set<Integer>>`

+ 0 pts Incorrect

2.3 **(no title)** **10** / 10 pts

+ 10 pts Correct

- ✔ **+ 1 pt** artists.stream()
- ✔ **+ 2 pts** filter on year = 2020 and genre is METAL
- ✔ **+ 3 pts** maxBy comparing weeksOnBillboardTop10
- ✔ **+ 1 pt** map(Artist::getName)
- ✔ **+ 2 pts** orElse("No artist found")
- ✔ **+ 1 pt** Result type is String

+ 0 pts Incorrect

Question 3

Multithreading **3** / 10 pts

3.1 **(no title)** **2** / 6 pts

+ 6 pts Correct

- ✔ **+ 2 pts** notify can be used if there are only two threads, one consumer and one producer. In this case, notifyAll degenerates to notify

+ 2 pts If notify is used when there are more than one producers or consumers, then only one thread can consume/produce at one time

+ 2 pts notifyAll is used for the more general situation where many threads consume, and many threads produce.

+ 0 pts Incorrect

3.2 **(no title)** **1** / 4 pts

+ 4 pts Correct

- ✔ **+ 1 pt** The implementation would work with if instead of while if there is only one producer and one consumer

+ 3 pts In get method, if a consumer thread T1 is waiting, another consumer thread T2 might come and consume the next prime when it becomes available. So by the time T1 resumes execution, there may not be a prime available. (Symmetric argument can be made for put method. An explanation for either get or put is sufficient, don't need both.)

+ 0 pts Incorrect

Question 4

Multithreading

9 / 24 pts

+ 24 pts Correct

- ✔ **+ 2 pts** T1 and T2 in WAITING state after issuing wait()
- ✔ **+ 3 pts** T1 and T2 both go to BLOCKED state on notifyAll from T3
- ✔ **+ 2 pts** T1 gets lock and goes to RUNNABLE (Alternatively T2 gets the lock and goes to RUNNABLE)
- ✔ **+ 2 pts** T1 gets to CPU, goes to RUNNING state, executes S1 (in above if T2 was picked, then T2 goes to RUNNING)

+ 3 pts T1 consumes T3's supply, exits S1 and goes to RUNNABLE state (if T2 was picked earlier then T2 consumes supply, exits S1 and goes to RUNNABLE state)

+ 2 pts T2 gets the lock (alternatively T1 gets the lock) and goes to RUNNABLE

+ 10 pts Different sequences possible between T1 and T2 that involved RUNNING, possibly BLOCKED, and WAITING

+ 0 pts Incorrect

Question 5

Design Patterns **2.5** / 26 pts

5.1 **(no title)** **0.5** / 14 pts

+ 14 pts Correct

States

+ 2 pts Correct

+ 0.5 pts Class open

+ 0.5 pts Class full

+ 0.5 pts Class Cancelled

✔ **+ 0.5 pts** Registration Closed

+ 0 pts Incorrect

Transitions from ClassOpen state

+ 5.5 pts Correct

+ 2 pts --> ClassOpen : student registers and student count < maxStudents

+ 2 pts --> ClassFull : student registers and student count == maxStudents

+ 1 pt --> RegistrationClosed: date == semesterStart + 2 weeks

+ 0.5 pts --> ClassCancelled: class is cancelled

✔ **+ 0 pts** Incorrect

Transitions from ClassFull state

+ 4.5 pts Correct

+ 1 pt --> ClassFull: student drops and first in wait list is registered

+ 2 pts --> ClassOpen: student drops and wait list is empty

+ 1 pt --> RegistrationClosed: date = semesterStart + 2 weeks

+ 0.5 pts --> ClassCancelled: class is cancelled

✔ **+ 0 pts** Incorrect

+ 1 pt RegistrationClosed -> ClassCancelled: class is cancelled

+ 1 pt ClassCancelled -> no transitions out of this state

+ 0 pts Incorrect

5.2 **(no title)** **2** / 12 pts

+ 12 pts Correct

✔ **+ 2 pts** Defining the abstract superclass for all states

---

Course registration open class - Singleton part

+ 5 pts Correct

+ 1 pt Declaration of Singleton instance variable

+ 1 pt private constructor

+ 1 pt Header for static method to return instance

+ 1 pt creating new instance if none exists

+ 1 pt return singleton instance

+ 0 pts Incorrect

Course registration open class - transition processing method

+ 5 pts Correct

+ 1 pt Method header with event parameter

+ 1 pt event is transition that returns an instance of ClassOpen

+ 1 pt event is transition that returns an instance of ClassFull

+ 1 pt event is transition that returns an instance of RegistrationClosed

+ 1 pt event is transition that returns an instance of ClassCancelled

+ 0 pts Incorrect

+ 0 pts Incorrect

CS 213 Spring '24 : Final Exam

This is a CLOSED NOTES exam worth 110 points. Your score on this exam will be doubled, to be out of 220 points, for 22% of your course grade.

Name: Akshaj Kammari NetID: ak1990

1. **Streams - 20 pts (15+5)**

   All answers must start with a way to source a stream, followed by a single sequence of stream operations (including `collect`) or an `Optional` class method ONLY. For each answer, the result MUST be assigned to a named and typed variable, as for example, `Integer res = ...` You do *not* need to write any import statements, and you do *not* need to write fully qualified stuff like `java.util.stream.Stream.map(...)`, just `map(...)` is sufficient.

   (a) In a document named "doc.txt" count *distinct* words (case INsensitive) by length (e.g. 3 distinct words of length 4, 5 distinct words of length 3, etc.) Assume that a word is a sequence of non-space characters, and each line of the document has one or more words.

```
int numDistinct = Files.lines(paths.get(doc.txt))

map(I -> I.split(" "))
.flatMap(Array::stream)
.distinct()
.collect(groupingBy(String::length.count()));
```

   (b) For a course with n students, an n X 3 array, `int[][] arr`, holds scores on 3 exams. Each row holds the exam scores for one student, one exam per column. Write code to get the average of cumulative exam score, over all students. For instance, if student1 scored 50,60,65 in the three exams, and student2 scored 75,85,90, then the cumulative exam score for student1 is 175 and that for student2 is 250, and the resulting average is 212.5

```
int cumulativeAve = arr.Stream().collect(averagingInt());
```

2. **Streams - 30 pts (11+9+10)**

Same requirements as detailed up front in the previous streams question. Given:

```
public enum Genre {ROCK, POP, COUNTRY, METAL, HIPHOP}
public class Artist {
   ...
   public String getName() { ... }      public Genre getGenre() { ... }
   public int getYear() { ... }         public int weeksOnBillboardTop10() { ... }
   public int streamCount() { ... }  // number of times artist was streamed in the year
}
```

For the following questions, assume a pre-populated list of artists, `List<Artist> artists`.

(a) Get, in an array, names of the top 10 most streamed artists in 2023, ranked from 1st to 10th.

```
String[] top10 = artist.stream().filter(a -> a.getYear() = 2023)
    sorted(comparing(artists::stream(count)).map(a -> a::getYear())
    .limit(10) collect(toList());
```

(b) For each genre, which years had artists with at least 10,000,000 streams in that genre?

```
Genre> Map Set <Integer>> = artists.stream.filter(s -> s.stream
    (count() >= 10000000).collect(groupBy(Artist.Genre, mapping(
    Artist::getYear
```

(c) Which artist (any one artist, if tied) in the METAL genre spent the most number of weeks on the Billboard Top 10 in 2020? If no artist is found, the result should be "No artist found".

```
ring artist=artists.stream().filter(a -> a.getYear() = 2020)
. filter(b -> b.(Genre = "METAL"), maxBy(comparing(Artist::weeks
  OnBillboardTop10))
. map(Artist::getName).orElse("No artist found")
```

NetID: ak1990

3. **Multithreading - 10 pts (6+4)**

In class, we discussed a producer-consumer scenario with the following code in a `PrimeCentral` class:

```
public synchronized int get() {
   while (available == false) {
      try { wait(); }
      catch (...) { }
   }
   available = false; notifyAll();
   return prime;
}
```

```
public synchronized void put(int prime) {
   while (available == true) {
      try { wait(); }
      catch (...) { }
   }
   this.prime = prime; available = true;
   notifyAll();
}
```

(a) What would be the effect of replacing `notifyAll()` with `notify()`? Does it depend on the number of threads? Explain.

Yes, it would depend on the number of threads. 'notifyAll()' seems to update all threads, while 'notify()' applies to only a singular thread.

(b) Starting with the original code (with `notifyAll()`), if you replaced `while` with `if`, would the new code be equivalent to the old? Does it depend on the number of threads? Explain.

No, the new code would not be equivalent to the old. It would depend on the number of threads; 'if' would only check the first, but 'while' would check them all.

4. **Multithreading - 24 pts**

Class X has *synchronized* methods S1 and S2. Consider an instance `xinstance` of class X, consumer threads T1 and T2, and supplier thread T3, and the following sequence of actions on `xinstance`:

- Thread T1 enters method S1, and at some point issues a `wait()`, because T3 hasn't supplied yet
- Subsequently, thread T2 enters method S1, and at some point issues a `wait()`, because T3 hasn't supplied yet
- Finally, thread T3 enters method S2, fills the supply, issues `notifyAll()`, finishes and exits S2

Assume that T3's supply is sufficient for only one of the consumers, and that once T3 has finished executing S2 as above, it is terminated and will not return to S2 again. But T1 and T2 are insatiable and will want to keep coming back to method S1 for more. Assume that there is no attempt by the application to safely terminate either T1 or T2, they are allowed to keep running.

List ONE plausible sequence of **thread states** that T1 and T2 will go through, starting from the time they issue a `wait()`, as described above, up to the time when there will not be any more state changes. For the purpose of this question, assume that there is an additional state, RUNNING, for when a thread is actually executing on the CPU. For each state change of each of T1 and T2, specify the cause of change of state.

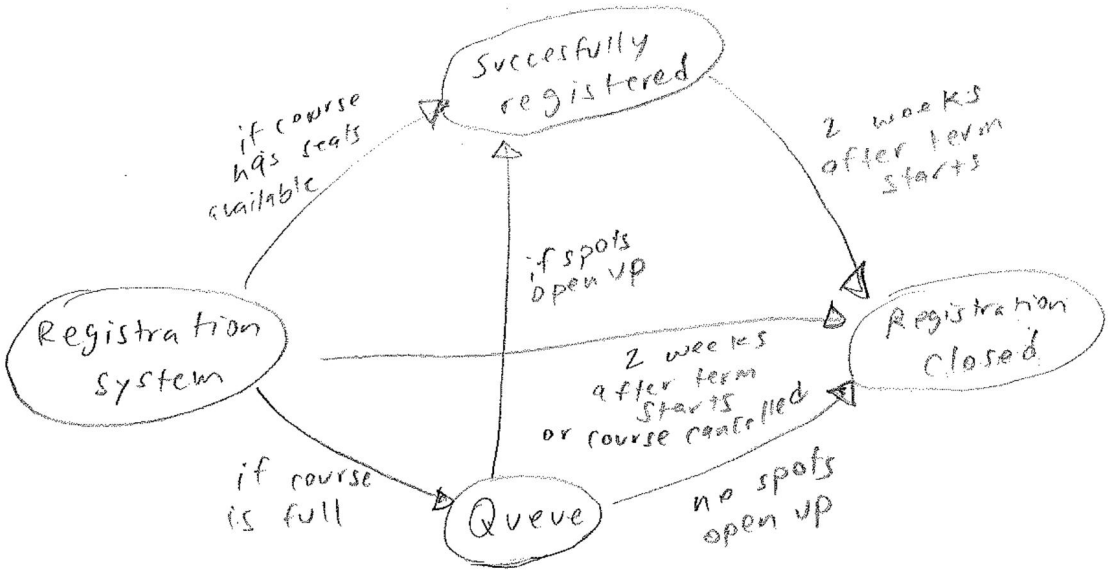
Because threads T1 and T2 are synchronized consumer threads, that means only 1 of them can be running at a time. T1 gets a wait from S1, but once T3 issues a notifyAll(), and exits S2, it opens availability for T1 or T2 for the synchronized methods. As either T1 or T2 goes into the running state the other

NetID: qk1990

5. **Design Patterns - 26 pts (14+12)**

Consider a course registration system. At a certain time before the start of the semester, the course is open for registration. A student may register for that course if there are seats available. If the course is full, students who try to register will be put on a wait queue, to be added in a first-come first-serve manner if spots open up. Two weeks after the start of the term, registration will be closed, which means nobody can either drop or add the course via this system. Note that a course may be cancelled without reason at any time during the registration period, or when registration is closed.

(a) Draw a state transition diagram for a single course registration. The nodes of the diagram should be the registration states (name them clearly) and the transition arrows between them (as applicable) should be labeled with the event that results in the transition.



(b) Show how your state transition diagram would be implemented using the State design pattern (which includes the Singleton pattern). First, write the *headers* of all the Java classes that you would need to implement the pattern. Next, fill in the class fields and method *headers* in (i) the class that corresponds to the course registration open state, and (ii) in any superclass that may be required by the pattern - you are not required to write class fields and method headers for for any other class. Lastly, for each method in course registration open class that handle state transitions, fill in the method body with the logic for the transitions–you may write the logic in pseudocode, so long as there is enough detail to translate to Java code.

Write your answer to 5(b) on this page

public abstract class courseRegistration {

public static courseRegistration state;