Q1 Object Design
10 Points

Consider a college where a faculty member could be a teacher or a researcher, or both. You are asked to design the most appropriate object-oriented relationship between faculty member, teacher, and researcher. Outline a Java object-oriented implementation with minimal class members: at least one field that distinguishes instances of a class from instances of other classes, one constructor, and one method in each class. You may NOT use abstract classes or interfaces. Credit will be proportional to the appropriateness of your design for this scenario. You are NOT asked to draw a UML diagram.

```
public class Faculty{
int facultyId;
String name, jobType;
Faculty(int id, String n, String job){
facultyId = id;
name = n;
jobtype = job;
}
}
public void showInfo(){
System.out.println("ID: "+facultyId);
System.out.println("Name: "+name);
System.out.println("Type: "+jobtype);
}
}

public class Teacher extends Faculty
{
String subject;
Teacher(int id, String n, String type, String sub)
{
super(id,n,type);
subject=sub;
}
void showTeacherInfo()
```

```
{
super.showInfo();
System.out.println("Subject: "+subject);
}
}
public class Researcher extends Faculty
{
String subject;

Researcher(int id, String n, String type, String sub)
{
super(id,n,type);
subject=sub;
}
void showResearcherInfo()
{
super.showInfo();
System.out.println("Subject: "+subject);
}
}

class TestClasses
{
public static void main(String[] args)
{
System.out.println("₩nTeacher: ");
System.out.println("------------------------------");
Teacher teacher=new Teacher(101, "Tim", "Teacher", "Computer
Science");
teacher.showTeacherInfo();

System.out.println("₩n₩nResearcher: ");
System.out.println("------------------------------");
Researcher researcher=new Researcher(201, "Mike",
"Researcher", "Search Engine Optimization");
researcher.showResearcherInfo();
System.out.println("₩n");
}
}
```

Q2 UML Class Diagram
20 Points

You are modeling a small part of an online flight reservation system using object-oriented design, according to the following description. A flight is a single non-stop hop between a pair of cities. A booking can include several flights and travelers (think of a family vacation hitting multiple cities), with the requirement that all the travelers are on all the flights in a booking.

Furthermore, every booking has a single owner who is one of the travelers on that booking (think one family member booking the tickets for the entire family), and the owner can manage functionality on the system that other travelers cannot. A separate ticket is issued and priced individually for each traveler on a booking, and the ticket applies to all flights within that booking.

"Draw" (see below for special drawing instructions) the most appropriate UML class diagram for your object-oriented design. Identify entities (classes/abstract classes/interfaces) clearly, and separate them according to their functionality--the separation should be faithful to the description above. Set up relationships between entities as precisely as possible. Inside each class in the diagram, only list the class name and minimal number of attributes required to characterize objects of that class (no operations needed). For the attributes, you are not required to show access level.

Instructions for how to draw (Do NOT upload pictures):

You must show all drawing in text, as described below.

Class/Interface:
Just list class name, and specify attribute names in that class in parentheses, e.g. Student (name, major)
Put down "abstract" or "interface" before entity name, if applicable

Relationships:

- Subclass: A <|--- B
- Subinterface, Interface implementation: same as subclass

- Association: ------ (with < or > on either side for direction)
- Aggregation: <>---- (for composition just write "composition" above or below the association line)
- Dependency: same as unidirectional association, but write "dependency" above or below the association line
- Association class: If A is association class for an X--Y association, simply
  spell it out instead of drawing a line hanging off the X--Y association. (e.g. "A is an association class for X--Y")
- Multiplicity: write above either end as in the UML problem set solutions

Write each relationship separately even if an entity happens to participate in more than one relationship - this way you don't need to do cumbersome "vertical" plain text drawings.
e.g. Student------Course
Student------Professor

Customer(name, email, phone number, address)
BookFlight(number, price, origin, destination, payment, billing)
Passenger(name, insurance, luggage, boarding)
ScheduleFlight(flightTimes, capacity, passengers, price)
Locations(origin, desitination)
Flight(number, departure, arrival, price)
Airport(name, location)

```
               1....*       1
Passenger <--------<> Customer
Customer is an association class of Passenger and
ScheduleFlight
            0...1          1...*
Customer --------->ScheduleFlight
             0...1           1...*
Passenger --------> ScheduleFlight
                          0....1         1.....*
ScheduleFlight <------------>Locations
                       1...*         1....*
ScheduleFlight <>-------->BookFlight
BookFlight is an association class of ScheduleFlight and Flight
               1....*            0....1
BookFlight ------------->Flight
```

```
        1....*              0.....1
Flight <--------------> Airport
```

Q3 Design Patterns
12 Points

You are developing an application for an online travel agency. The agency manages several packages to various tourist sites. Each package consists of transport to site (day 1), followed by activities for each of the next 3 days, and finally transport back home (day 5). The transport details and the actual daily activities differ depending on the package.

Show how your would use the template method design pattern to implement the various tour packages. Write classes and method headers as minimally needed (no logic needed inside method bodies) to clearly show the design pattern code structure.

```
abstract class Package{
 public boolean details;
 public abstract void day2();
 public abstract void day3();
 public abstract void day4();
 public final void TourCreated() {
         try {
System.out.print("Tour was successful");
}
catch (Exception e) {
System.out.print("Tour was created");
}
}

public final void plan(boolean details) {
day2();
day3();
if(details) {
TourCreated();
}
}
}
```

```
class Package1 extends Package{
@override
public void day2() {
System.out.println("Passenger transported to site on Day 1")
System.out.println("Day 2 plans created");
}
@override
public void day3() {
System.out.println("Day 3 plans created");
}
@override
public void day4() {
System.out.println("Day 4 plans created");
}
}

class Package2 extends Package{
@override
public void day2() {
System.out.println("Passenger transported to site on Day 1")
System.out.println("Day 2 version 2 plans created");
}
@override
public void day3() {
System.out.println("Day 3 version 2 plans created");
}
@override
public void day4() {
System.out.println("Day 4 version 2 plans created");
}
}
```

Q4 Streams
14 Points

All answers must start with a way to source a stream, followed by a single sequence of ONLY stream operations including collect and any operation on Optional.

For each answer, the result MUST be assigned to a named and typed variable:
e.g. **Integer res** = ... sequence of stream operations ...
Here **res** (name) is the result variable of **Integer** (type)

You don't need to write import statements for any of the classes and interfaces you use.

Q4.1
9 Points

In a document named **doc.txt**, count unique words (case **in**sensitive) by length, i.e. how many unique words of what length. (For instance, 3 words of length 4, 5 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.

```
Stream<String> stream = Files.lines(Paths.get("doc.txt"));
Map<Integer, Long> map = stream.flatMap(x ->
Arrays.stream(x.split(" ")))
.map(String::toLowerCase)
.collect(Collectors.toSet()).stream()
.collect(Collectors.groupingBy(String::length,Collectors.counting())
)
```

Q4.2
5 Points

Consider an nx2 array, double[][] arr, in which each of the 2 columns represents a *vector*. Write code to get the dot product of the column vectors, i.e. sum of products of corresponding items. For example,

[1,2,3].[2,2,4] = 1*2 + 2*2 + 3x4 = 18

```
Map<Integer, Long> map = Arrays.stream(arr)
.map(x -> x[0] * x[1])
.reduce(0, Integer :: sum));
```

Q5 Streams
21 Points

All answers to this question must start with a way to source a stream, followed by a single sequence of ONLY stream operations including collect and any operation on `Optional`.

For each answer, the result MUST be assigned to a named and typed variable:
e.g. **Integer res** = ... sequence of stream operations ...
Here **res** (name) is the result variable of **Integer** (type)

You don't need to write import statements for any of the classes and interfaces you use.

For this question, you are given an enumeration and a class as follows:

```
public enum Language {JAVA, PYTHON, C, RUBY, SCALA, PHP}

public class Coder {
  ...
  public String getName() { ... }
  public String getCity() { ... }
  public int getYear() { ... }
  public int getLOC() { ... }   // LOC => Lines Of Code
  public Language getLanguage() { ... }
}
```

Assume a pre-populated list of coders, `List<Coder> coders`. From this list, extract the data required in each of the following questions.

Q5.1
7 Points

Gather into a list the names of all coders for year 2020, sorted from most prolific (greatest LOC) to least

```
Comparator<Coder> LOCCompare =
Comparator.comparing(Coder::getLOC).reversed();
List<Coder> prolific = coders.stream()
```

```
.filter(s -> s.getYear().equals("2020"))
.sorted(LOCCompare).collect(Collectors.toList())
```

Q5.2
8 Points

For each language, which cities had coders that wrote at least 50,000 lines of code in that language?

```
List<String> cities = coders.stream()
.filter(s -> s.getLOC() >= 50000)
.map(x -> x.getCity())
.distinct().
.collect(Collectors.toList());
```

Q5.3
6 Points

Which coder (get the name of any one coder) wrote the most lines of code in Python in 2021? If no coder is found, the result should be "No coder found"

```
String pythonCoder = coders.stream()
.filter(s -> s.getLanguage() == Language.PYTHON)
.filter(s -> s.getYear()
.equals("2021")
.sorted(LOCCompare)
.map(x -> x.getName())
.reduce("No coder found", (ans, val) -> ans.equals("No coder
found"), val:ans))
```

Q6
20 Points

Consider the following **BankAccount** class:

```
public class BankAccount {

    private float balance;
    public BankAccount(float money) { balance = money; }

    public void deposit(float money) {
        balance = balance + money;
    }

    public void withdraw(float money) throws Exception {
       if (balance >= money) balance = balance - money;
       else throw new Exception(money +
                              " is more than balance");

    public float getBalance() { return balance;}
}
```

Q6.1
8 Points

Detail one scenario in which concurrent use of a **BankAccount** object by two threads results in an incorrect bank balance when the threads are done executing. Show the exact interleaved sequence of operations executed by the threads. No partial credit - either the scenario is correct, or it isn't.

Thread A: retrieve balance
Thread B: retrieve balance
Thread A: Increment retrieved value by money, result is balance + money
Thread B: Decrement retrieved value by money, result is balance - money
Thread A: Store result in balance, balance is now balance + money
Thread B: Store result in balance, balance is now balance - money
Thread A's result is lost, overwritten by Thread B

Q6.2
4 Points

Show how you would modify the **BankAccount** class to make it *thread safe*, i.e. multiple threads can concurrently access an instance without the above problem. Keep in mind that you should not overdo it by forcing unnecessary sequentiality among concurrent threads.

```
public class BankAccount {
private float balance;
public BankAccount(float money) { balance = money; }
public void deposit(float money) {
public synchronized (balance) {
lock();
balance = balance + money;
unlock();
}
}
public void withdraw(float money) throws Exception {
public synchronized (balance) {
lock();
if (balance >= money) {
balance = balance - money;
}
else {
throw new Exception(money +
" is more than balance");
}
unlock();
}
}
public float getBalance() { return balance;}
}
```

Q6.3
8 Points

Would your thread safe class of 6.2 above offer sufficient protection against concurrently running threads, one of which is transferring money from one account to another, and another is accessing one or both accounts for
withdrawal or deposit or balance check? If so, describe how thread safety is ensured. If not, outline a scenario in which the balance shown is incorrect, and briefly explain how it could be corrected

(no need to write code for the fix, a clear and concise explanation will do). No partial credit.

The above class would not always protect against those two actions. If you were to access the same bank account, one trying to receive money in a transfer and one looking to withdrawal, the balance may not be correct when whichever comes second begins to run or reach a deadlock. You could lock the transfer thread and once the deposit/check/withdrawal is done executing, you can notify the transfer thread, unlock, then run().

Q7 Multithreading
15 Points

Class X holds a data buffer, and has *synchronized* methods C (for consuming from buffer) and S (for supplying to buffer). 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 C, and at some point issues a `wait()`, because T3 hasn't supplied yet
- Subsequently, thread T2 enters method C, and at some point issues a `wait()`, because T3 hasn't supplied yet
- Finally, thread T3 enters method S, fills in the supply, issues a `notifyAll()`, finishes and exits S

Assume that T3's supply is sufficient for *only one* of the consumers, and that once T3 has finished executing S as above, it is terminated (i.e. it will not return to S again).

But T1 and T2 are insatiable and will want to keep coming back to method C for more. Assume that there is no attempt by the application to safely terminate either T1 or T2, i.e. they are allowed to keep running.

List ONE plausible sequence of thread states that T1 and T2 might 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.

Thread - Method - State - cause
T1 S1 wait - T3 hasn't supplied yet
T2 S1 wait - T3 hasn't supplied yet
T3 S2 Running - supply filled up
T1 S1 Running - woken up by notifyall
T1 Other Running - T1 starts working on an item read
T2 S1 Running - T1 existed S1, T2 enters
T2 Other Running - T2 starts working on an item read
T1 S1 Running - T1 enters sync method and uses the unused S1
T2 S1 wait - waiting from T1 sync method
T1 Other Running - T1 starts item read
T2 S1 Running - T1 exited S1, T2 enters
T2 Other Running - T2 starts an item read
...
T1 S1 wait - T3's supply is exhausted
T2 S1 wait - T3's supply is exhausted

T1 and T2 will repeatedly come back to the synchronized method S1 (only one of them being allowed at a time since it is a synchronized method) until all items produced by S2 are consumed. T1 and T2 will then wait indefinitely (since no more items will be produced by T2) until the process is terminated.

Q8 Inheritance
13 Points

Although saying "Cylinder IS A Circle" doesn't make sense, it can be implemented in Java using inheritance in a legitimate way. Implement **Circle** and **Cylinder** classes with a minimal number of fields, constructors, and methods required to demonstrate all aspects of legitimate inheritance. You must fill in logic for all constructors and methods. If needed, you may assume a Point class that holds x and y coordinates of a 2D point.

```
class Circle {
int radius;
Circle(int r) {
radius = r;
}
public int getRadius() {
return radius;
}
public void setRadius(int radius) {
this.radius = radius;
}
double area() {
return 3.14 * radius * radius;
}
}

class Cylinder extends Circle {
int height;
Cylinder(int r, int h) {
super(r);
height = h;
}
public int getHeight() {
return height;
}
public void setHeight(int height) {
this.height = height;
}
double volume() {
return area() * height;
}
}
```

Final Exam ● Graded

Student
KANGHWI LEE

Total Points
95.5 / 125 pts

Question 1	Object Design	6 / 10 pts
Question 2	UML Class Diagram	13 / 20 pts
Question 3	Design Patterns	8 / 12 pts
Question 4	Streams	12 / 14 pts
4.1	(no title)	8 / 9 pts
4.2	(no title)	4 / 5 pts
Question 5	Streams	12.5 / 21 pts
5.1	(no title)	5.5 / 7 pts
5.2	(no title)	2.5 / 8 pts
5.3	(no title)	4.5 / 6 pts
Question 6	(no title)	20 / 20 pts
6.1	(no title)	8 / 8 pts
6.2	(no title)	4 / 4 pts
6.3	(no title)	8 / 8 pts
Question 7	Multithreading	13 / 15 pts
Question 8	Inheritance	11 / 13 pts