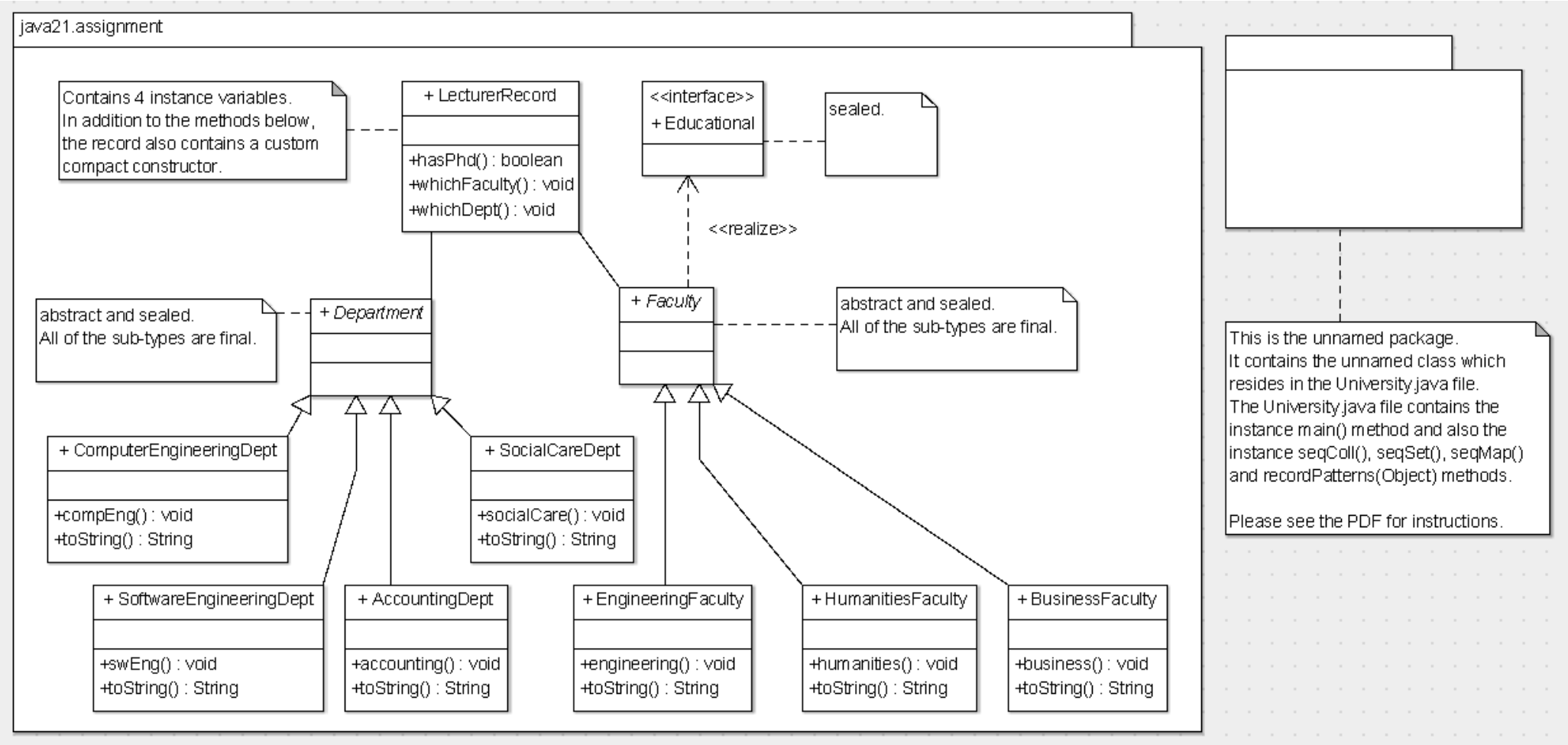
Java 21 Assignment Instructions



Note: As this is <u>not</u> a UML course, the above diagram is an overview. Also, for those of you for which English is not you first language and/or you are not familiar with Universities, a Faculty contains several Departments.

This is a successor to the Java 17 assignment. If you have not done the Java 17 assignment, please do it before this one, **BUT** make all classes/interfaces are *public* – it will make this Java 21 assignment easier.

Refactoring since Java 17 assignment: The fact that this Java 21 assignment uses an unnamed class and an instance *main()* method as an entry point, has several implications for the Java 17 implementation:

- The UML *University* class from the Java 17 assignment no longer exists. The file *University.java* is retained but is re-factored to be an unnamed class with an instance *main()* entry point.
    - note that an unnamed class cannot reside in a named package so the "*package assignment*" in University.java (from Java 17) must be removed/commented out.
- As the unnamed class is now in an unnamed package, this poses 2 issues when accessing classes in named packages:
    - to access classes in named packages, an unnamed class must *import* them (or namespace qualify them). A general import *java21.assignment.*;* is cleanest.
    - in order to *import* them, the classes (in the named packages) must be *public* and not package-private:
        - as many of the classes/interfaces in the Java 17 assignment were package-private this requires refactoring – as you can now see in the UML, **all of the classes/interfaces are now *public***
        - if you are modifying the Java 17 assignment then you need to create separate files for each class/interface (as the filename must match the name of the *public* class/interface).
        - the default constructors for these classes will follow the access on the class, so they will be *public*
- Given these changes, I took the opportunity to rename the package to "*java21.assignment*".

The following instructions will help in coding the assignment:

- *University*
    - this is an unnamed class with an instance *main()* method
    - in the *main()* method do the following:
        - invoke the *seqColl()* method – see the method for instructions.
        - invoke the *seqSet()* method – see the method for instructions.
        - invoke the *seqMap()* method – see the method for instructions.
        - create a *LecturerRecord* for "Mike Bloggs"; who is 44; works in the software engineering department in the engineering faculty.
            - you will need instances of *SoftwareEngineeringDept* and *EngineeringFaculty* when creating your *LecturerRecord*.
        - invoke the method *recordPatterns()* passing the reference down. Sample output: As he is only 44, nothing will be output.
        - create a *LecturerRecord* for "Alan Austin"; who is 64; works in the accounting department in the business faculty.
            - you will need instances of *AccountingDept* and *BusinessFaculty* when creating your *LecturerRecord*.

- invoke the method *recordPatterns()* passing the reference down. Sample output:

```
Name: Alan Austin,
Age: 64,
Faculty: Business,
Department: Accounting
```

- create a *LecturerRecord* for "Lisa Bloggs"; who is 65; works in the social care department in the humanities faculty.
    - you will need instances of *SocialCareDept* and *HumanitiesFaculty* when creating your *LecturerRecord*.
- invoke the method *recordPatterns()* passing the reference down. Sample output:

```
Name: Lisa Bloggs,
Age: 65,
Faculty: Humanities,
Department: Social Care
```

- in the *seqColl()* method do the following:
    - create a *SequencedCollection* typed for *LecturerRecord* implemented by an *ArrayList*
    - create *LecturerRecord*'s for the following staff who all work in the software engineering department in the engineering faculty:
        - the lecturers name is "Jane Bloggs" and she is 24.
        - the lecturers name is "Dr. Anne Bloggs" and she is 35.
        - the lecturers name is "Joe Bloggs PhD" and he is 54.
    - add Jane to the front of the collection
    - add Anne to the front of the collection
    - add Joe to the end of the collection
    - output the collection

```
[LecturerRecord[name=Dr. Anne Bloggs, age=35, faculty=Engineering,
dept=Software Engineering], LecturerRecord[name=Jane Bloggs, age=24,
faculty=Engineering, dept=Software Engineering], LecturerRecord[name=Joe Bloggs
PhD, age=54, faculty=Engineering, dept=Software Engineering]]
```

    - retrieve the first element in the collection

```
getFirst() : LecturerRecord[name=Dr. Anne Bloggs, age=35, faculty=Engineering,
dept=Software Engineering]
```

    - retrieve the last element in the collection

```
getLast() : LecturerRecord[name=Joe Bloggs PhD, age=54, faculty=Engineering,
dept=Software Engineering]
```

- remove the last element from the collection

removeLast() : LecturerRecord[name=**Joe** Bloggs PhD, age=54, faculty=Engineering, dept=Software Engineering]

- output the collection again

[LecturerRecord[name=Dr. **Anne** Bloggs, age=35, faculty=Engineering, dept=Software Engineering], LecturerRecord[name=**Jane** Bloggs, age=24, faculty=Engineering, dept=Software Engineering]]

- using an enhanced-for loop, process the collection from the beginning to the end

LecturerRecord[name=Dr. **Anne** Bloggs, age=35, faculty=Engineering, dept=Software Engineering]
LecturerRecord[name=**Jane** Bloggs, age=24, faculty=Engineering, dept=Software Engineering]

- using an enhanced-for loop, process the collection from the end to the beginning

LecturerRecord[name=**Jane** Bloggs, age=24, faculty=Engineering, dept=Software Engineering]
LecturerRecord[name=Dr. **Anne** Bloggs, age=35, faculty=Engineering, dept=Software Engineering]

- in the *seqSet()* method do the following:
  - create a *SequencedSet* typed for *LecturerRecord* implemented by a *LinkedHashSet*
  - create *LecturerRecord*‘s for the following staff who all work in the accounting department in the business faculty:
    - the lecturers name is “Jane Austin” and she is 24.
    - the lecturers name is “Dr. Charlotte Bronte” and she is 35.
    - the lecturers name is “Anne Bronte PhD” and she is 54.
  - add Jane 3 times to the front of the collection
  - add Charlotte to the front of the collection
  - add Jane to the end of the collection
  - add Anne to the end of the collection
  - output the collection

[LecturerRecord[name=Dr. **Charlotte** Bronte, age=35, faculty=Business, dept=Accounting], LecturerRecord[name=**Jane** Austin, age=24, faculty=Business, dept=Accounting], LecturerRecord[name=**Anne** Bronte PhD, age=54, faculty=Business, dept=Accounting]]

- retrieve the first element in the collection

getFirst() : LecturerRecord[name=Dr. **Charlotte** Bronte, age=35, faculty=Business, dept=Accounting]

- retrieve the last element in the collection

getLast() : LecturerRecord[name=**Anne** Bronte PhD, age=54, faculty=Business, dept=Accounting]

- remove the first element from the collection

removeFirst() : LecturerRecord[name=Dr. **Charlotte** Bronte, age=35, faculty=Business, dept=Accounting]

- output the collection again

[LecturerRecord[name=**Jane** Austin, age=24, faculty=Business, dept=Accounting], LecturerRecord[name=**Anne** Bronte PhD, age=54, faculty=Business, dept=Accounting]]

- using an enhanced-for loop, process the collection from the beginning to the end

LecturerRecord[name=**Jane** Austin, age=24, faculty=Business, dept=Accounting]
LecturerRecord[name=**Anne** Bronte PhD, age=54, faculty=Business, dept=Accounting]

- using an enhanced-for loop, process the collection from the end to the beginning

LecturerRecord[name=**Anne** Bronte PhD, age=54, faculty=Business, dept=Accounting]
LecturerRecord[name=**Jane** Austin, age=24, faculty=Business, dept=Accounting]

- in the *seqMap()* method do the following:
  - create a *SequencedMap* whose keys are *LecturerRecord*'s and whose values are *String*'s. Use a *LinkedHashMap* implementation.
  - create *LecturerRecord*'s for the following staff who all work in the social care department in the humanities faculty:
    - the lecturers name is "King Lear" and he is 88.
    - the lecturers name is "Goneril" and she is 55.
    - the lecturers name is "Regan" and she is 50.
    - the lecturers name is "Cordelia" and she is 45.
  - add Goneril to the front of the collection; she maps to "Eldest".
  - add Regan to the front of the collection; she maps to "Middle".
  - add Cordelia to the end of the collection; she maps to "Youngest".
  - add King Lear to the end of the collection; he maps to "Father".

- output the collection

{LecturerRecord[name=**Regan**, age=50, faculty=Humanities, dept=Social Care]=Middle, LecturerRecord[name=**Goneril**, age=55, faculty=Humanities, dept=Social Care]=Eldest, LecturerRecord[name=**Cordelia**, age=45, faculty=Humanities, dept=Social Care]=Youngest, LecturerRecord[name=**King Lear**, age=88, faculty=Humanities, dept=Social Care]=Father}

- retrieve the first entry in the collection

firstEntry() : LecturerRecord[name=**Regan**, age=50, faculty=Humanities, dept=Social Care]=Middle

- retrieve the last entry in the collection

lastEntry() : LecturerRecord[name=**King Lear**, age=88, faculty=Humanities, dept=Social Care]=Father

- remove (poll) the last entry from the collection

pollLastEntry() :LecturerRecord[name=**King Lear**, age=88, faculty=Humanities, dept=Social Care]=Father

- output the collection again

{LecturerRecord[name=**Regan**, age=50, faculty=Humanities, dept=Social Care]=Middle, LecturerRecord[name=**Goneril**, age=55, faculty=Humanities, dept=Social Care]=Eldest, LecturerRecord[name=**Cordelia**, age=45, faculty=Humanities, dept=Social Care]=Youngest}

- using an enhanced-for loop, process the collection from the beginning to the end; output both the keys and values.

LecturerRecord[name=**Regan**, age=50, faculty=Humanities, dept=Social Care]; Middle
LecturerRecord[name=**Goneril**, age=55, faculty=Humanities, dept=Social Care]; Eldest
LecturerRecord[name=**Cordelia**, age=45, faculty=Humanities, dept=Social Care]; Youngest

- using an enhanced-for loop, process the collection from the end to the beginning; output both the keys and values.

LecturerRecord[name=**Cordelia**, age=45, faculty=Humanities, dept=Social Care];
Youngest
LecturerRecord[name=**Goneril**, age=55, faculty=Humanities, dept=Social Care];
Eldest
LecturerRecord[name=**Regan**, age=50, faculty=Humanities, dept=Social Care];
Middle

- in the *recordPatterns(Object obj)* method do the following:
  - algorithm: assuming that the retirement age is 65, calculate the staff that either should be retired or are reaching retirement age within the next year i.e. age >= 64.
  - using a *switch* expression that returns a *String* for output in a *System.out.println()* do the following:
    - using a record pattern in the *case* label, provide a guard (on the right of the *when* clause) that ensures only lecturers that are >= 64 years of age are selected:
      - if the lecturer is >= 64 years of age then using a text block and the *formatted()* method, build up a string for output; the string should contain the name, age, faculty and department of the lecturer.
      - *yield* the string back from the *switch* expression to the *System.out.println()*.
    - if the *case* label matches *null* or no other match can be made (*default*), return the empty string "" to the *System.out.println()*. Do this on one line of code.