CS2030 Practical Assessment #1

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

CS2030 Practical Assessment #1

To manage the surge in the number of Covid-19 cases, the government mandates that residents who are exposed or tested positive follow Protocol 1-2-3. A modified version of the protocol is presented below:

- Protocol 1 — for individuals who are feeling unwell

  For high-risk individuals,

  - if you test negative, you will follow up with your doctor.
  - if you test positive, you will remain under Protocol 1 with daily testing.

  For low-risk individuals,

  - if you test negative, you will follow up with your doctor.
  - if you test positive, you will continue with Protocol 2.

- Protocol 2 — for individuals who are assessed as low-risk and tested positive

  You should self-isolate for 72 hours, and are required to take a test everyday.

  At the end of 72 hours,

  - if you test negative, you can exit self-isolation.
  - if you continue to test positive, continue self-isolation under Protocol 2 and self-test everyday until you obtain a negative result or after 7 days for vaccinated individuals (after 14 days for unvaccinated individuals), whichever is earlier.

- Protocol 3 — for close contacts of a Covid-19 positive case

  Individuals would have received a Health Risk Notice (HRN) for 5-day monitoring. You should take a self-test daily.

  - if you test negative, you can continue normal activities. You must complete the 5-day monitoring period.
  - if you test positive, you will continue with Protocol 1 for high-risk individuals (or Protocol 2 for low-risk individuals).

Each person is associated with an integer health risk score, as well as a vaccination status. The health risk score ranges from 0 to 10 with a score of 8 and above assessed as high-risk; a score of 7 and below is assessed as low-risk. The vaccination status comprises a string of vaccines with each letter representing one vaccine. For example, "PP" is two doses of Pfizer, "M" represents a dose of Moderna, "SSM" represents three doses (two doses of Sinovac and a Moderna booster). An empty string represents no vaccination. Moreover, a person is considered vaccinated if he/she has been administered two or more doses of vaccine of any type.

Task

Your task is to track the history of each Covid-19 case, which includes the current protocol as well as Covid-19 test history.

Take note!

There are altogether five levels. You are required to complete ALL levels.

You should keep to the constructs and programming discipline instilled throughout the module.

- Write each `class`/`interface` in its own file.
- Ensure that ALL object properties and class constants are declared `private final`.
- Ensure that that your classes are NOT cyclic dependent.
- You are NOT allowed to use java libraries, other than those from `java.lang`, `java.util.List` and `java.util.Comparator`.
- Use only the `ImList` class provided with this question.
- You are NOT allowed to use java keywords/literals: `null`, `instanceof`
- You are NOT allowed to use Java reflection, i.e. `Object::getClasses` and other methods from `Class`
- You are NOT allowed to use `*` wildcard imports.
- You are NOT allowed to define anonymous inner classes, or lambdas.
- You are NOT allowed to define array constructs, e.g. `String[]` or using ellipsis, e.g.`String...`

You may assume that all tests provide valid arguments to methods; hence there is no need to validate method arguments.

Level 1

Write a class `Person` with a constructor that takes in an identification as a `String`, the vaccination status as a `String`, followed by an integer health risk score.

Include two boolean methods `isVaccinated` and `isHighRisk`, as well as a `toString` method that returns a string representation of a person. Note that high-risk is denoted with a caret `^`, and low-risk denoted with lowercase `v`.

```
jshell> new Person("A123", "PP", 8)
$.. ==> A123/PP/^

jshell> new Person("A123", "PP", 8).isVaccinated()
$.. ==> true

jshell> new Person("A123", "PP", 8).isHighRisk()
$.. ==> true

jshell> new Person("B234", "M", 7)
$.. ==> B234/M/v

jshell> new Person("B234", "M", 7).isVaccinated()
$.. ==> false

jshell> new Person("B234", "M", 7).isHighRisk()
$.. ==> false

jshell> new Person("Z999", "", 1)
$.. ==> Z999//v

jshell> new Person("Z999", "", 1).isVaccinated()
$.. ==> false

jshell> new Person("Z999", "", 1).isHighRisk()
$.. ==> false
```

Level 2

There are currently two types of Covid-19 tests: `ART` and `PCR`

- ART

  An Antigen Rapid Test is administered that produces one of three possible results. If the result is `C`, the outcome is negative; if the result is `CT`, the outcome is positive. However, if the result is `T`, the outcome is invalid.

- PCR

  A polymerase chain reaction test uses primers to match a segment of the virus's genetic material. Rather than dwell into the intricacies of genetic matching, we replace with string matching instead. You may assume that

there are only four "strains" of Covid-19: `alpha`, `beta`, `delta` and `omicron`. If the result of a PCR test matches any of these, the outcome is positive; otherwise the outcome is negative. Note that a PCR test is always valid.

Include two boolean methods `isPositive` and `isValid`, as well as a `toString` method that returns a string representation of the test which comprises the type of test (`A` for ART; `P` for PCR), followed by `+` if positive, `-` if negative, or uppercase `X` if invalid.

```
jshell> Test test = new ART("C")
test ==> A-

jshell> test.isValid()
$.. ==> true

jshell> test.isPositive()
$.. ==> false

jshell> test = new ART("CT")
test ==> A+

jshell> test.isValid()
$.. ==> true

jshell> test.isPositive()
$.. ==> true

jshell> test = new ART("T")
test ==> AX

jshell> test.isValid()
$.. ==> false

jshell> test.isPositive()
$.. ==> false

jshell> test = new PCR("omicron")
test ==> P+

jshell> test.isValid()
$.. ==> true

jshell> test.isPositive()
$.. ==> true

jshell> test = new PCR("decepticon")
test ==> P-

jshell> test.isValid()
$.. ==> true

jshell> test.isPositive()
$.. ==> false
```

Level 3

A potential Covid-19 case is initiated whenever a person starts feeling unwell and falls under Protocol 1.

Write a class `Case` with a constructor that takes in a `Person` followed by a `PCR` test and creates a default case with Protocol 1.

Include a `test` method that takes in any type of Covid-19 test, and determines if the case should remain under Protocol 1, or discharged with an advisory. In this level, you may assume that the person is a high-risk individual.

Include a `toString` method that returns a string representation of the case comprising of the person, the history of test results, and the current protocol. Note that invalid tests are not recorded.

The history of test results should be stored in an immutable list, so that the string representation of the list would automatically include the square brackets `[..]`. You should not hardcode the square brackets as it would deem to have violated the restriction on using the array construct.

Note that a discharged case can continue to be tested and have the tests tracked. As soon as a test becomes positive, the person will be advised to seek medical attention, even if subsequent tests are negative.

Include the `getCurrentProtocol` and `getTestHistory` methods to return the current protocol and the list of tests. Although this violates the "Tell-Don't-Ask" principle, we do it for ease of testing your program.

Use the following definition of the `Protocol` interface to help you define protocol `P1`.

```
interface Protocol {
    Protocol next(Person person, Test test); // generates the next protocol
}
```

Another useful tip is to avoid using "case" as a variable or method name in your program as "case" is a Java keyword.

```
jshell> new Case(new Person("A123", "PP", 8), new ART("C"))
|  Error:
|  incompatible types: ART cannot be converted to PCR
|  new Case(new Person("A123", "PP", 8), new ART("C"))
|                                        ^----------^

jshell> Case c = new Case(new Person("A123", "PP", 8), new PCR("alpha"))
c ==> A123/PP/^ [P+] P1

jshell> Protocol p = c.getCurrentProtocol()
p ==> P1

jshell> p instanceof P1
$.. ==> true

jshell> c = c.test(new PCR("beta"))
c ==> A123/PP/^ [P+, P+] P1

jshell> c = c.test(new ART("T")) // invalid test not recorded
c ==> A123/PP/^ [P+, P+] P1

jshell> c = c.test(new ART("CT"))
c ==> A123/PP/^ [P+, P+, A+] P1

jshell> c = c.test(new ART("C"))
c ==> A123/PP/^ [P+, P+, A+, A-] Discharged: follow up with doctor

jshell> c = c.test(new ART("C"))
c ==> A123/PP/^ [P+, P+, A+, A-, A-] Discharged: follow up with doctor

jshell> c = c.test(new ART("CT"))
c ==> A123/PP/^ [P+, P+, A+, A-, A-, A+] Discharged: seek medical attention

jshell> c = c.test(new ART("C"))
c ==> A123/PP/^ [P+, P+, A+, A-, A-, A+, A-] Discharged: seek medical attention

jshell> c.getTestHistory()
$.. ==> [P+, P+, A+, A-, A-, A+, A-]

jshell> c.getTestHistory() instanceof ImList
$.. ==> true
```

Level 4

Now modify the `Case` class to cater to low-risk individuals. Specifically, if a low-risk individual is tested positive, he/she will be placed under Protocol 2, `P2`.

There is a 3-day isolation period within which the case cannot be discharged even if a test is negative. An individual with a positive test will continue in Protocol 2 until after 7-days (vaccinated) or 14-days (unvaccinated), after which the case will be discharged irrespective of the test result. You may assume that one valid test is administered each day.

The `next` method of the `Protocol` interface should now be defined as:

```
interface Protocol {
    Protocol next(Person person, Test test, int numOfDays);
}
```

Do not modify the interface.

The following depicts a sample run for a low-risk and vaccinated individual. Your program needs to work for unvaccinated individuals as well.

```
jshell> new Case(new Person("B234", "M", 7), new PCR("decepticon"))
$.. ==> B234/M/v [P-] Discharged: follow up with doctor
```

```
jshell> Case c = new Case(new Person("B234", "M", 7), new PCR("delta"))
c ==> B234/M/v [P+] P2

jshell> Protocol p = c.getCurrentProtocol()
p ==> P2

jshell> p instanceof P2
$.. ==> true

jshell> c.test(new PCR("zero"))
$.. ==> B234/M/v [P+, P-] P2

jshell> c = c.test(new PCR("zero")).test(new ART("C"))
c ==> B234/M/v [P+, P-, A-] P2

jshell> c.test(new ART("C")) // discharged after 72 hours (or 3 days)
$.. ==> B234/M/v [P+, P-, A-, A-] Discharged: complete

jshell> c.test(new ART("C")).test(new ART("C"))
$.. ==> B234/M/v [P+, P-, A-, A-, A-] Discharged: complete

jshell> c.test(new PCR("beta"))
$.. ==> B234/M/v [P+, P-, A-, P+] P2

jshell> c.test(new PCR("beta")).test(new PCR("ok"))
$.. ==> B234/M/v [P+, P-, A-, P+, P-] Discharged: complete

jshell> Test pt = new ART("CT")
pt ==> A+

jshell> Case c = new Case(new Person("C345", "MM", 7), new PCR("delta"))
c ==> C345/MM/v [P+] P2

jshell> c = c.test(pt)
c ==> C345/MM/v [P+, A+] P2

jshell> c = c.test(pt)
c ==> C345/MM/v [P+, A+, A+] P2

jshell> c = c.test(pt)
c ==> C345/MM/v [P+, A+, A+, A+] P2

jshell> c = c.test(pt)
c ==> C345/MM/v [P+, A+, A+, A+, A+] P2

jshell> c = c.test(new ART("T")) // invalid test
c ==> C345/MM/v [P+, A+, A+, A+, A+] P2

jshell> c = c.test(pt)
c ==> C345/MM/v [P+, A+, A+, A+, A+, A+] P2

jshell> c = c.test(pt)
c ==> C345/MM/v [P+, A+, A+, A+, A+, A+, A+] P2

jshell> c = c.test(pt) // Discharged after 7-days even if tested positive
c ==> C345/MM/v [P+, A+, A+, A+, A+, A+, A+, A+] Discharged: complete

jshell> c = c.test(new ART("C"))
c ==> C345/MM/v [P+, A+, A+, A+, A+, A+, A+, A+, A-] Discharged: complete

jshell> c.test(pt)
$.. ==> C345/MM/v [P+, A+, A+, A+, A+, A+, A+, A+, A-, A+] Discharged: seek medical attention

jshell> c.test(pt).test(pt)
$.. ==> C345/MM/v [P+, A+, A+, A+, A+, A+, A+, A+, A-, A+, A+] Discharged: seek medical attention

jshell> c.test(pt).test(pt).test(new ART("C"))
$.. ==> C345/MM/v [P+, A+, A+, A+, A+, A+, A+, A+, A-, A+, A+, A-] Discharged: seek medical atten
```

Level 5

A Health Risk Notice (HRN) is served to individuals who have been identified as contact cases under Protocol 3, `P3`.

Write a class `ContactCase` with a constructor that takes in a person, a valid Covid-19 test, and an existing Covid-19 case as the source of contact.

Individuals under Protocol 3 have to undergo 5-day monitoring. If there is a positive test within this period, he/she will continue with Protocol 1 (high-risk) or Protocol 2 (low-risk).

The following depicts a sample run for a low-risk individual who has received a Health Risk Warning. Your program needs to cater to high-risk individuals as well.

```
jshell> Case c = new Case(new Person("A123", "PP", 8), new PCR("omicron"))
c ==> A123/PP/^ [P+] P1

jshell> new ContactCase(new Person("B234", "M", 7), new ART("CT"), c)
$.. ==> B234/M/v [A+] P2

jshell> Test t = new ART("C")
t ==> A-

jshell> Case d = new ContactCase(new Person("B234", "M", 7), t, c)
d ==> B234/M/v [A-] P3

jshell> Protocol p = d.getCurrentProtocol()
p ==> P3

jshell> p instanceof P3
$.. ==> true

jshell> d = d.test(t)
d ==> B234/M/v [A-, A-] P3

jshell> d = d.test(t)
d ==> B234/M/v [A-, A-, A-] P3

jshell> d = d.test(t)
d ==> B234/M/v [A-, A-, A-, A-] P3

jshell> d = d.test(t)
d ==> B234/M/v [A-, A-, A-, A-, A-] P3

jshell> d = d.test(t)
d ==> B234/M/v [A-, A-, A-, A-, A-, A-] Discharged: complete

jshell> d = d.test(t)
d ==> B234/M/v [A-, A-, A-, A-, A-, A-, A-] Discharged: complete

jshell> d = d.test(new PCR("delta"))
d ==> B234/M/v [A-, A-, A-, A-, A-, A-, A-, P+] Discharged: seek medical attention
```

Lastly, we would like to find how one case is linked to another, i.e. it's lineage. Where appropriate, include the `getLineage` method that returns an immutable list `ImList<Case>` with the first element as the earliest source of infection.

The following is a continuation of the sample run.

```
jshell> void printLineage(ImList<Case> lineage) {
   ...>     boolean first = true;
   ...>     for (Case c : lineage) {
   ...>         System.out.println((first ? "" : "-> ") + c);
   ...>         first = false;
   ...>     }
   ...> }
|  created method printLineage(ImList<Case>)

jshell> printLineage(c.getLineage())
A123/PP/^ [P+] P1

jshell> printLineage(d.getLineage()) // c -> d
A123/PP/^ [P+] P1
-> B234/M/v [A-, A-, A-, A-, A-, A-, A-, P+] Discharged: seek medical attention

jshell> ContactCase e = new ContactCase(new Person("C345", "", 10), new ART("CT"), d)
e ==> C345//^ [A+] P1

jshell> printLineage(e.getLineage()) // c -> d -> e
```

```
A123/PP/^ [P+] P1
-> B234/M/v [A-, A-, A-, A-, A-, A-, A-, P+] Discharged: seek medical attention
-> C345//^ [A+] P1
```