# MACOVID: A System for Tracking COVID Infections in Massachusetts

e.g., PANTRACK: A System for Tracking Pandemics in Iceland

# JOSEPH K PARSHLEY III

# Assignment 1 --1/12/2021 7:08 AM

For this assignment, you will describe and implement release 1 of your term project. You will incorporate *an abstract class*, *inheritance*, *upcasting or downcasting*, and *polymorphism*. You are free to choose a project that interests you but if you prefer, the instructor and your facilitator will be happy to suggest a topic. If you are already an experienced developer, this is an opportunity to build a challenging application (check with your facilitator if it requires significant API’s) or discuss research with the instructor. It is OK to name a project with much more scope than you can accomplish in the course (as in the example above): we will not hold you to completing everything associated with it. What we do expect is that you specify and implement a set of do-able requirements within such scope.

Submit this completed Word document. Replace as indicated. Please observe and retain the gray text. Your materials—in black 12-point Times New Roman—should not exceed 5 pages excluding references, figures, and appendices. Use the Appendix sections for additional material if you need to. These will be read only on an as-needed basis.

We want you to develop in Eclipse preferably or else IntelliJ (talk to your facilitator about exceptions). As you code, use JUnit tests whenever possible but certainly by week 2—package-by-package, class-by-class, and method-by-method, except for trivial methods and those requiring I/O. Use non-Junit classes for testing the latter. Keep the evaluation criteria in mind, listed at the end.

For this assignment, you do not need to read data from a file—you can build all data into the code.

Include a ReadMe file describing where to run the application from, and including necessary execution notes. All JUnit tests will be assumed runnable.

# 1.1 SUMMARY DESCRIPTION *EVALUATION CRITERION (i) APPLIES*

One- or two-paragraph overall description of your proposed term project—half-page (12-point Times New Roman) limit. By the end, term projects will incorporate most of the techniques discussed in the course. To do this, you may need to alter the direction of your project or introduce an additional project in future. You may alter this or even replace it as the semester progresses. You will probably find it useful to use your project acronym.

This project concerns a system for people to investigate the rates of infection of the cities and towns for the state of Massachusetts. To use the system, called MACOVID, the user would submit a file of COVID test results for the towns and cities of Massachusetts. The system would display a table in the console sowing the test result as well as the percent infected along with which phase the state is in (Red, Yellow, Green or White). The focus of the term project is the taking the file of raw test data and automatically categorizing the cities and town into its COVID phase as well as the trends and percentages of that data.

## 1.2 I/O EXAMPLE FROM *PROJECTED* COMPLETED PROJECT *EVALUATION CRITERION (i) APPLIES*

Provide an example of projected *concrete* output for designated input. You will not be held to fulfilling exactly this—it is just explanatory at this point, to indicate where your project is going. We recognize that project direction and details will change as the term progress. This section refers to the project as a whole, not just to what you will produce this week, so we can gain an idea of what you have in mind overall.

Here is an example of a simple MACOVID test.

**MACOVID**:

Please enter the filename of the test results you are interested in analyzing.

**User**: testResults.csv

**System**:

With the given test results, here is the latest up to date COVID data:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| City/Town | Pop. | Case Count | | Average Daily Incidence Rate per 100,000 | Infection Trend | Tests | | | Percent Positive | Change in present positivity |
|  |  | Total | Last 14 days | (Last 14 Days |  | Total | Total Last 14 days | Positive |  |  |
| Andover | 63500 | 772 | 193 | 75.6 | Rising |  |  | 1125 | 10.40 | Higher |
| Boston | 63500 | 772 | 193 | 25.6 | Rising |  |  | 1125 | 4.9 | No Change |
| Boxford | 63500 | 772 | 193 | 28.7 | Falling |  |  | 1125 | 2.88 | Lower |
| Groveland | 63500 | 772 | 193 | 0 | No change |  |  | 1125 | 0 | No change |
| Haverhill | 63500 | 772 | 193 | 0 | No change |  |  | 1125 | 0 | No change |
| Ipswich | 63500 | 772 | 193 | 0 | No change |  |  | 1125 | 0 | No change |
| Lawrence | 63500 | 772 | 193 | 0 | No change |  |  | 1125 | 0 | No change |
| Merrimack | 63500 | 772 | 193 | 0 | No change |  |  | 1125 | 0 | No change |
| Methuen | 63500 | 772 | 193 | 0 | No change |  |  | 1125 | 0 | No change |
| North Andover | 63500 | 772 | 193 | 0 | No change |  |  | 1125 | 0 | No change |

## 1.3 REQUIREMENTS IMPLEMENTED IN THIS RELEASE *EVALUATION CRITERION (ii) APPLIES*

Supply [functional requirements](https://docs.google.com/document/d/1eU7eINLDxmrf793D4OF2yGT4ry_SW3GQGoVDYzecGHc/edit?usp=sharing) statement that you accomplished for this assignment, i.e., functionality that the application provides for the user. Please state requirement in declarative form, as illustrated in the examples, because here we want to know the functionality intended (*what*, not *how*). For example, the following is *not* a proper functional requirement: *TicTac will have a class for O’s and a class for X’s.* It is common to mistake design elements like this for requirements. To get started, state what the application will accept as input, like requirement 1.3.1 below.

Keep in mind that the implementation of your requirements will incorporate *an abstract class*, *inheritance*, *upcasting or downcasting*, and *polymorphism*; that will probably influence the requirements you choose to implement in this assignment. The example material supplied should be deleted before you submit.

### 1.3.1 Accept State Testing Data

MACOVID outputs to the console a prompt for the user to enter the name of the file holding the tests data.

The user supplies the name of the local file consisting of data in tabular form. The data fields in this file consists of:

* City/Town name
* Population
* Total Case
* Two Week Case Count
* Total Tests
* Two Week Total Tests
* Total Positive Tests
* Testing Rate
* Testing Results Date

### 1.3.2 Report Testing Results for each Municipality

MACOVID reports the data of each of the current stats entered along with a table of the history table data to console. The system outputs top console a success message saying the analysis is complete and the results are located in the file [filename]. The results filename will consists of the date, a descriptive string and file extension. An example filename could be “20210404MACOVID\_Results.txt”. The data fields in this file consist of fields existing in the original test data file along with :

* City/Town name
* Population
* Total Case
* Two Week Case Count
* Average Daily Incidence Rate per 100,000 (Last 14 Days)
  + Used to delineate the Color Coding (Grey, Green, Yellow, Red)

|  |  |  |  |
| --- | --- | --- | --- |
| Population | | | |
| **Group** | **< 10k** | **10k < 50k** | **50k <** |
| Grey | <= 10 total cases | <= 10 total cases | <= 15 total cases |
| Green | <= 15 total cases | <10 avg cases/100k AND >10 total cases | <10 avg cases/100k AND > 15 total cases |
| Yellow | <= 25 total cases | >=10 avg cases/100k OR >= 5% pos rate | >=10 avg cases/100k OR >= 4% pos rate |
| Red | > 25 total cases | >=10 avg cases/100k AND >= 5% pos rate | >=10 avg cases/100k AND >= 4% pos rate |

* Color
* Change in Last Week
* Total Tests
* Two Week Total Tests
* Total Positive Tests
* Percent Positivity
* Testing Rate
* Testing Results Date
* Relative Change in Case Counts
  + Number of new cases occurring over the current two-week period compared to the previous two-week period.
* Total Tests

## 1.4 ILLUSTRATIVE OUTPUT FROM IMPLEMENTATION *EVALUATION CRITERION (ii) APPLIES*

### Provide illustrative output from your implemented application (so far) showing that the requirements have been met. Explain what class.method(s) produce it.

The following is produced by *main().*

*src/macovid/testResults.csv* is the filepath to the sample test results downloaded from the web.

Enter the test filename:

src/macovid/testResults.csv

I am troubleshooting why all the values are null/0.

null 0 0 0 0 0 0 0.00 null

null 0 0 0 0 0 0 0.00 null

null 0 0 0 0 0 0 0.00 null

null 0 0 0 0 0 0 0.00 null

null 0 0 0 0 0 0 0.00 null

null 0 0 0 0 0 0 0.00 null

null 0 0 0 0 0 0 0.00 null

null 0 0 0 0 0 0 0.00 null

null 0 0 0 0 0 0 0.00 null

null 0 0 0 0 0 0 0.00 null

…

null 0 0 0 0 0 0 0.00 null

Process finished with exit code 0

## 1.5 YOUR DIRECTORY

### Show a screenshot of your directory. This should include a parallel directory of JUnit tests where possible—package-by-package, class-by-class, and method-by-method, except for trivial ones.

All methods in the MACOVID example are trivial so far except for ones that perform displays, and these are not suitable for JUnit testing.

Graphical user interface, text

Description automatically generated

## 1.6 TECHNIQUES IMPLEMENTED *EVALUATION CRITERION (iii) APPLIES*

Your implementation should include *inheritance*, *polymorphism*, and *either an abstract class or interface* at least once, and in a manner that is useful to your application. Explain where and how you applied these, using the headings below.

### 1.6.1 Class model and Sequence Diagram

Identify where you included *inheritance*, *polymorphism*, and *abstract classes* or *interfaces* in your class model. Make classes and members *static* or not as per their intended usage. To do this use tools (e.g., Visio and Lucidchart), PowerPoint, or a combine models as in [this example](https://docs.google.com/spreadsheets/d/1vBmDVtWWh3EX0oehFFLRU0P6eR-fn4d0qVg1-XOUooM/edit?usp=sharing) (which you are free to cut and paste from). Insert indications in red (as in [this example](https://docs.google.com/spreadsheets/d/1ZvkerE9FkWHWwVGdzuy7YMBU6oBMFGZbA4sotFETs8Y/edit?usp=sharing)) to show where the three features below apply.

The class model for the MACOVID example is [here](https://docs.google.com/spreadsheets/d/1ecb0iHRARiyd2LsCW8WLOKyaiJ-WH-9TdMczW2OaLzI/edit?usp=sharing). *Phase* is abstract. Polymorphism occurs in the versions of *displayPhase*(). The figure includes inheritance.

### 1.6.2 Code showing an abstract class or interface

Show the relevant code (only) and explain why an abstract class or interface is appropriate here. It should be clear where the code is located (class and method).

There are only four kinds of phases, so there is no need for the *Phase* base class to be concrete since there will be no generic Phases.

package macovid;

/\*\*

\* Phase: The category of infection rate for the city or town based on population

\* There are 4 phases

\* - Red: - Greater than 25 total cases for population under 10k

\* - Greater than or equal to 10 average cases/100k AND

\* greater than or equal to 5% positivity rate for 10k - 50k population

\* - Greater than or equal to 10 average cases/100k AND

\* greater than or equal to 5% positivity rate for over 50k population

\* - Yellow: - Less than or equal to 25 total cases for population under 10k

\* - Greater than or equal to 10 average cases/100k OR

\* greater than or equal to 5% positivity rate for 10k - 50k population

\* - Greater than or equal to 10 average cases/100k OR

\* greater than or equal to 5% positivity rate for over 50k

\* - Green: - Less than or equal to 15 total cases for population under 10k

\* - Less than 10 average cases/100k AND

\* greater than 10 total cases for 10k - 50k population

\* - Less than or equal to 10 average cases/100k AND

\* greater than 15 total cases for over 50k

\* - Grey: - Less than or equal to 10 total cases for population under 10k

\* - Less than or equal to 10 total cases/100k for 10k - 50k population

\* - Less than or equal to 15 total cases/100k rate for over 50k

\*/

public abstract class Phase {

protected String color = "n/a";

public Phase(){}

public Phase(String colorName){

color = colorName;

}

public String getColor() {

return color;

}

// displayPhase is used to show the color of the determined phase in the test results

public abstract void displayPhase();

}

### 1.6.3 Code showing polymorphism

Show the relevant code (only) and explain why *polymorphism* is appropriate here. Recall that polymorphism is implemented in one of two ways – overriding methods in subclasses or overloading methods in the same class where the method signatures are different – and allowing the language runtime to dynamically invoke the correct method. It should be clear where the code is located (class and method).

The *Phase* class uses the abstract method *displayPhase()* which will be used to display the color of the phase that the city or town is under. This is polymorphism. The relevant code is shown below.

package macovid;

public class PhaseGreen extends Phase{

private String color = "Green";

@Override

// Use polymorphism to display the phase color name based on which phase class // is used

**public void displayPhase() {**

**System.out.printf("Your state is in Phase %s%n.", this.color);**

**}**

}

### 1.6.4 Code showing upcasting or downcasting

Show the relevant code (only) and explain why upcasting or downcasting is appropriate here. It should be clear where the code is located (class and method).

This portion was solely added to satisfy the assignment criteria. It displays all the phases.

package macovid;

import java.util.ArrayList;

public class PhaseGreen extends Phase{

private String color = "Green";

private ArrayList<Phase> thePhases = new ArrayList<Phase>();

public PhaseGreen(String color) {

super(color);

}

@Override

// Use polymorphism to display the phase color name based on which phase class is used

public void displayPhase() {

System.out.printf("Your state is in Phase %s%n.", this.color);

}

public void displayAllPhases(){

// Postcondition: All phases are displayed per Phase.displayPhase()

// display the current phase

displayPhase();

// display the other phases

for (Phase phase: thePhases){

if (phase instanceof Phase) {

**((Phase)phase).displayPhase();**

} else {

phase.displayPhase();

}

}

}

}

## 1.7 YOUR CODE

Unless your facilitator requests another method, copy your Eclipse project to your file system, zip it, and attach it. Please contact your facilitator in advance if you want to request an alternative means.

## 1.8 EVALUATION OF ASSIGNMENT 1



## Appendix 1 (if needed; should be referenced above, and will be read as-needed only)

## Appendix 2 (if needed; should be referenced above, and will be read as-needed only)

. 1/28/2021 7:03 AM

# Assignment 3

Implement the next release of your term project (preferably, or start a new one if you have to). You will incorporate generics. The same instructions as in Assignment 2 apply to this completed Word document, the gray text, the 5 page limit, appendices, JUnit tests, and a ReadMe file.

## 3.1 SUMMARY DESCRIPTION, UPDATED AS NECESSARY

*EVALUATION CRITERION (i) APPLIES*

One- or two-paragraph overall description of your proposed term project. Color in red the parts different from Assignment 2.

Your response replaces this.

## 3.2 ADDITIONAL REQUIREMENTS (FEATURES) IMPLEMENTED IN THIS RELEASE

*EVALUATION CRITERION (i) APPLIES*

Title and one or two sentences per requirement. Don’t repeat requirements implemented for prior assignments unless they are necessary to provide context—in which case, make it clear which are new vs. old.

### 3.2.1 Your title replaces this. (OLD / NEW REQUIREMENT)

Your response replaces this.

### 3.2.2 Your title replaces this. (OLD / NEW REQUIREMENT)

Your response replaces this.

### 3.2.3 Your title replaces this. (OLD / NEW REQUIREMENT)

Your response replaces this.

### 3.2.4 Your title replaces this. (OLD / NEW REQUIREMENT)

Your response replaces this.

### 3.2.5 ….

## 3.3 I/O EVIDENCE OF ACCOMPLISHING THE REQUIREMENTS LISTED ABOVE

*EVALUATION CRITERION (ii) APPLIES*

Provide an example of actual input / output corresponding to the requirements above

Your response replaces this.

### Input File(s)

File ….

### Input / Output

Console I/O:

### Output File(s)

File …

## 3.4 YOUR DIRECTORY

Show a screenshot of your directory. This should include a parallel directory of JUnit tests where applicable—package-by-package, class-by-class, and method-by-method, except for trivial and inapplicable ones.

Your response replaces this.

## 3.5 YOUR UPDATED CLASS MODEL AND CLARIFICATION OF HOW THE EXECUTION WORKS

*EVALUATION CRITERION (i) APPLIES*

Supply a main use case, the class model, and the sequence diagram corresponding to the use case. These should be consistent and clear. Indicate clearly in your class model where you applied generics. To do this use tools, PowerPoint, or a combine models as in [this example](https://docs.google.com/spreadsheets/d/1vBmDVtWWh3EX0oehFFLRU0P6eR-fn4d0qVg1-XOUooM/edit?usp=sharing) (which you are free to cut and paste from). Insert indications in red to show where generics apply.

Your response replaces this.

## 3.6 WHERE GENERICS ARE IMPLEMENTED

*EVALUATION CRITERION (iii) APPLIES*

### 3.6.1 Class model fragment showing generic class

Explain where and how you applied *generic classes* in your class model.

Your response replaces this.

### 3.6.2 Code (including test code), input (if applicable), and output showing generics

Explain why the use of *generics* is appropriate here.

Your response replaces this.

## 3.7 YOUR CODE

*EVALUATION CRITERION (iii) APPLIES*

Unless your facilitator arranges another method, copy your Eclipse project to your file system, zip it, and attach it. Please contact your facilitator in advance if you want to request another transmission process (e.g., github).

## 3.8 INSTRUCTOR’S EVALUATION

