Version: 28-Apr-2023 11:30 PM

CS 2223 D23 Term. Homework 4

Homework Instructions

- This homework is to be completed individually. If you have any questions as to what constitutes improper behavior, review the examples as I have posted online http://web.cs.wpi.edu/~heineman/html/teaching_/cs2223/d23/#policies.
- Due Date for this assignment is on canvas.
- Submit your assignments electronically using the canvas site for CS2223. Login to canvas.wpi.edu and locate HW4. You must submit a single ZIP file that contains all of your code as well as the written answers to the assignment.
- All of your Java classes must be defined in a packager USERID where USERID is your CCC user id.
- Submission information is found at the end of this document.

Getting Started

This homework assignment is about problem solving. Using the data structures and algorithms presented over the past few weeks (including the final weeks of this class), this assignment gives you the opportunity to take the ideas presented in class and solve problems.

You will work with several graph algorithms and graph data structures.

- Copy **`algs.hw4.Histogram`** into **`USERID.hw4`**
- Copy **`algs.hw4.map.FlightMapApp`** into **`USERID.hw4`** and rename as **`Delta`**
- Copy **`algs.hw4.map.FlightMapApp`** into **`USERID.hw4`** and rename as **`Southwest`**
- Copy **`algs.hw4.map.FilterLower48`** into **`USERID.hw4`**
- Copy **`algs.hw4.map.MapSearch`** into **`USERID.hw4`**

You will create the following new classes in **`USERID.hw4`**

- Create a class **`USERID.hw4.Overlap`**
- Create a class **`USERID.hw4.Connected`**
- Create a class **`USERID.hw4.Hub`**
- Create a class **`USERID.hw4.FlightStats`**
- Create a class **`USERID.hw4.LongestOfShortest`**

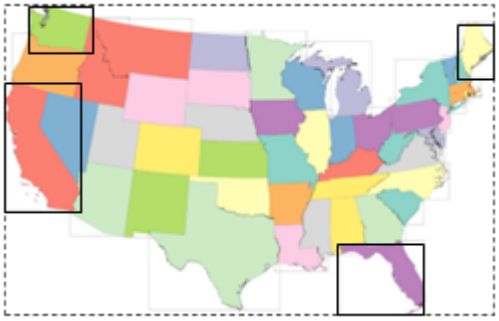
Version: 28-Apr-2023 11:30 PM

Working with GPS

I provide an **`algs.hw4.map.GPS`** class that will prove quite useful in this assignment. It records a GPS location through two **`float`** values, a longitude and a latitude.

First compute a "bounding box" in terms of longitude and latitude. The lower 48 states in the United States can be "roughly" approximated by constructing a large bounding box that encloses these four states. You will use this computation later.

State	Minimum Longitude	Minimum Latitude	Maximum Longitude	Maximum Latitude
California	-124.41	32.53416	-114.131	42.00952
Maine	-71.0839	42.97776	-66.9499	47.45969
Florida	-87.6349	24.5231	-80.0314	31.00089
Oregon	-124.566	41.99179	-116.464	46.29204



Yes, I know the lines are not "straight" when drawn on the globe, but just work with me!

Version: 28-Apr-2023 11:30 PM

Airport Application Domain

The first problem is concerned with flight networks for two major airlines, **Delta** and **Southwest Airlines**. Between December 19th and December 28th, 2022, Southwest Airlines canceled more than 16,000 flights. Even though the initial reason was a massive winter storm, the airline was unable to "restart" its flying schedule for nearly a week after the weather had cleared up. While we won't be able to use the existing data to identify the root cause of the Southwest failure, these two airline networks provide a useful domain for graph algorithms.

I provide two files – **delta.json** and **southwest.json** – that contain information about each airport connection for each airline that I downloaded on January 2nd, 2023. Each direct flight that is possible (from **`airport1`** to **`airport2`**) contains the following data:

```
{
  "airport1": {
    "country": "United States", "iata" : "ABQ", "icao" : "KABQ",
    "lat":35.040218, "lon":-106.609001, "name": "Albuquerque International Airport"
  },
  "airport2":{
    "country": "United States", "iata" : "AUS", "icao" : "KAUS",
    "lat":30.194521, "lon":-97.6698, "name": "Austin Bergstrom International Airport"
  }
}
```

Use the *icao* field (International Civil Aviation Organization) to uniquely identify each airport, which is located at a specific GPS location with designated latitude (**`"lat"`**) and longitude (**`"lon"`**). Note that longitude values may be negative because Greenwich, England contains the Prime Meridian, and a negative value declares the GPS location is to the west of Greenwich, England.

Each airport has a "country", because sometimes an airline has a direct flight from the United States to a neighboring country, such as "Mexico." My class **`FlightMap`** provides all the information you need.

Highway Transportation Application Domain

Smart phones can help direct us while driving because of countless information about roads that has been encoded. I have download highway information from https://travelmapping.net/graphs/, a helpful web site with loads of information. Each TMG file contains information like below. My class **`HighwayMap`** provides all the information you need.

```
TMG 2.0 traveled
10796 12986                                  [(# of Vertices) (# of Edges)]
I-95(86)/MA113 42.817377 -70.915375          [GPS coordinate of intersection]
I-95(88)/MA110 42.845073 -70.902758
…
985 141 MA62                                 [labeled edge between two vertices]
143 144 MA60
…
```

Version: 28-Apr-2023 11:30 PM

Q1 Histogram [10 pts.]

One frequent tool for analyzing data is a Histogram that records the frequency of specific key values. In addition, a histogram allows you to group together ranges of keys to provide aggregate information.

Copy **`algs.hw4.Histogram`** into your **`USERID.hw4`** package and modify its **`main()`** method to produce the following output, which shows the raw output (in order) together with two summarized outputs using different bin sizes.

```
Raw Histogram             [FIXED! My earlier output was incorrect]
0     2
1     2
2     1
4     2
7     1
8     1
9     1
11    1
13    2
14    1
15    3
17    2
19    1
Histogram (binSize=1)
0-0   2
1-1   2
2-2   1
3-3   0
4-4   2                   [Note:  When a histogram has a binSize, it]
5-5   0                   [should always start reporting from 0,]
6-6   0                   [regardless of minimum key that it records.]
7-7   1
8-8   1
9-9   1
10-10 0
11-11 1
12-12 0
13-13 2
14-14 1
15-15 3
16-16 0
17-17 2
18-18 0
19-19 1
Histogram (binSize=5)
0-4   7
5-9   3
10-14 4
15-19 6
```

Q1.1 Bonus Question [1pt]: Create a method **`int slidingMaximum(width)`** that returns the low value of the range (low, low + width) for which the sum total of values associated with these keys is a

maximum. For the sample table above, **`slidingMaximum(5)`** should return **13**, since the sum of the values associated with the keys (13, 14, 15, 16 and 17) is 8, a maximum for this data collection. Modify your **`main()`** method to demonstrate it works.

Q2 Delta and Southwest Flight Networks

Copy the **`algs.hw4.map.FlightMapApp`** class into **`USERID.hw4`** package and rename the class to be **`Delta`**; copy a second time, this time renaming to **`Southwest`**.

Within each of these new class files, be sure to modify the **`main()`** method to load up either "delta.json" or "southwest.json" as appropriate. Run each of these classes so you can visualize their flight networks. However, the results are not that easy to understand, so you have some work to do.

Q2.1 Implement FilterLower48 [5 pts.]

Copy the **`algs.hw4.map.FilterLower48`** class into your **`USERID.hw4`** and modify it to only accept airports that are in the country "United States" and whose GPS coordinates are within the bounding box of allowed GPS coordinates as identified on page 1 of this homework description.

You know your **`FilterLower48`** class is correct when you can determine that:

- The dataset contains 289 airports for Delta Airlines, but only 187 in the United States
- The dataset contains 125 airports for Southwest Airlines, but only 99 in the United States

And when you run **Delta** you might be able to see the contours of the United States (especially the major hub for Delta Airlines in Atlanta, Georgia).

For all subsequent questions in this assignment, it is assumed that you will filter all airports to be within the lower 48 states.

Q2.2 Determine airports with Southwest Flights but no Delta Flights [10 pts.]

Create a class **`USERID.hw4.Overlap`** that determines those United States airports in the Lower 48 states that are served by Southwest Airlines **but not by Delta Airlines**.

Your program must print out a list of the ICAO codes for the airports. Print each ICAO code one per line in any order. Hint: There are six of them.

Bonus Question 2.2.1 [1pt] : Which of these airports is also a Space Port with a valid Commercial Space Launch Site license from the FAA?

Version: 28-Apr-2023 11:30 PM

Q2.3 Determine Connectivity in an Airline Network [10 pts.]

Is it possible to use a single airline carrier (either Delta or Southwest Airlines) and **start from Boston Logan Airport** (whose ICAO code is "KBOS") and fly to every other airport (using any number of connecting flights) in that carrier's network by only using that carrier?

To fly from Boston to Highfill, Arkansas (ICAO code "KXNA"), one possible route would be:

- fly from KBOS to KTPA (Tampa, Florida);
- to KSLC (Salt Lake City, Utah);
- to KTUS (Tucson, Arizona);
- to KMSP (Minneapolis, St. Paul);
- to KXNA.

One of these airlines (either Delta or Southwest) has two airports in its network (AIRPORT-1 and AIRPORT-2) that you cannot reach from Boston.

Create a class **`USERID.hw4.Connected`** that prints out (a) the name of the airline; and (b) the ICAO codes for these two airports. Your program must process the graphs to produce the following output:

```
The name of the airline is XXXXX
The airports that cannot be reached from KBOS using XXXXX are:
AIRPORT-1
AIRPORT-2
```

Q2.4 Hub and Spoke Airline Models [10 pts.]

From Wikipedia: "The hub-and-spoke system allows an airline to serve fewer routes, so fewer aircraft are needed. The system also increases passenger loads; a flight from a hub to a spoke carries not just passengers originating at the hub, but also passengers originating at multiple spoke cities."

From Wikipedia: The United States airport system was point-to-point, controlled by CAB, until deregulation late 1960s/early 1970s, and eventually the 1978 Airline Deregulation Act when they switched to the hub concept.

Let's arbitrarily define a *hub* to be an airport that is connected to more than 75 other airports by a direct flight. Create a class **`USERID.hw4.Hub`** that prints for each airline the ICAO code for each of its hubs together with the number of flights at that hub (this list does not need to be in any specific order). Your output should be something like the table on the right:

```
DELTA
KXXX        ##
KYYY        ##
KZZZ        ##
…

SOUTHWEST
KXXX        ##
KYYY        ##
KZZZ        ##
…
```

Hint: One of these airlines only has a single hub assuming you are restricting all airport information to the lower48 as requested earlier.

Version: 28-Apr-2023 11:30 PM

Q2.5 Find the shortest and longest direct flight based on distance [10 pts.]

For both Delta and Southwest airlines, what is the longest direct flight in their network? Or the shortest direct flight? Compute the distance by using the **`GPS.distance(GPS other)`** method, which computes the distance between two GPS coordinates. For this question, you **must truncate** distance values into integers, like **`distance = (int) gps1.distance(gps2).`**

Create a class **`USERID.hw4.FlightStats`** that prints out (a) the name of the airline; (b) the ICAO codes for the shortest and longest flights for each airline; and (c) the average flight distance for all of the airline's flights. NOTE: Continue to filter out airports using the Lower48 filter from before.

```
Shortest flight for Delta is from ???? to ???? for ??? miles
Longest flight for Delta is from ???? to ???? for ??? miles
Average Delta flight distance = 786.9050847457627          [Note: check your work]

Shortest flight for Southwest is from ???? to ???? for ??? miles
Longest flight for Southwest is from ???? to ???? for ??? miles
Average Southwest flight distance = 817.312762973352       [Note: check your work]
```

Then for each airline compute a **Histogram** (recall question 1 on this assignment) for the flight distances of each airline and output the results using a **`binSize`** of 500 miles.

These histograms should look like this (with small formatting variations, especially on first row). Naturally do not print out my annotated comment, but instead replace all ??? with real values.

```
Delta Airlines:
Histogram (binSize=500)

0-499        ???
500-999      ???
1000-1499    ???
1500-1999    ???
2000-2499    26          [Note: I include these two so you can check your work]
2500-2999    4

Southwest Airlines:
Histogram (binSize=500)

0-499        ???
500-999      ???
1000-1499    ???
1500-1999    60          [Note: I include these two so you can check your work]
2000-2499    4
```

Version: 28-Apr-2023 11:30 PM

Q2.6 Find Longest of Shortest possible trips between any two airports [15 pts.]

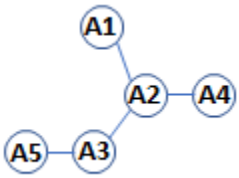
Create a class **`USERID.hw4.LongestOfShortest`** that solves the "All Pairs, Shortest Distance" over the network graphs for each carrier and determines the maximal of these shortest paths – that is, a pair of airports for which the total accumulated distance (when trying to compute the shortest path between these airports) is **larger than any other pair of airports in the network**.

For each carrier (Delta or Southwest), given any two airports with a direct flight within its network, construct a weighted undirected digraph (using the **`AdjMatrixEdgeWeightedDigraph`** class), where the weight of the edge representing that connection is the computed distance (in miles) between the GPS locations of the two airports. Note: Be sure to filter using Lower48 as before.

Use **Floyd-Warshall** to solve the "All Pairs, Shortest Distance" for this edge-weighted digraph. Based on the results, compute the shortest distance (and the actual path) between any two vertices in the graph. Now you need to find the largest value from among all possible pairs. This maximal value reflects the worst case, that is, the most work performed when minimizing the travel from any point A to point B.

The "`test.json`" sample provides specific guidance on this question. In this airport network, there are five airports (A1 to A5) whose GPS distances from each other are:

Actual	A1	A2	A4	A3	A5
A1	0	533.293	0	0	0
A2	533.293	0	576.334	694.805	0
A4	0	576.334	0	0	0
A3	0	694.805	0	0	861.994
A5	0	0	0	861.994	0



In this graph, the computed "shortest path" between two airports using **Floyd Warshall** is reported below. For example, the shortest travel distance from **A1** to **A4** is 533.2933 + 694.8053 = 1228.0986 and that is reflected in table below. The "longest" shortest path entry is between **A4** and **A5.**

Flight	A1	A2	A4	A3	A5
A1	0	533.293	1109.627	1228.099	2090.092
A2	533.293	0	576.334	694.805	1556.799
A4	1109.627	576.334	0	1271.139	2133.133
A3	1228.099	694.805	1271.139	0	861.994
A5	2090.092	1556.799	2133.133	861.994	0

The column and row labels are not strictly in ascending order. This is not important and reflects the fact that it can be hard to impose a meaningful ordering on the labels for vertices in a graph.

To compute the average flight-to-distance ratios, you need to accumulate the *efficiency* of each shortest possible distance. The table below computes efficiency (Flight / Actual). The average of all efficiency values is 3.004437757217313.

Efficiency	A1	A2	A4	A3	A5
A1	0	1.000	1.693	1.480	14.108
A2	1.000	0	1.000	1.000	3.598
A4	1.693	1.000	0	1.021	4.144
A3	1.480	1.000	1.021	0	1.000
A5	14.108	3.598	4.144	1.000	0

Version: 28-Apr-2023 11:30 PM

For the test airport network, the resulting output should look something like this:

```
Test : Total Flight Distance is 2133.133159485895 but airports are only
514.7274553433999 miles apart.
KA4 -> KA2 for 576.3340228351461              [+4 for path of “longest” shortest path]
KA2 -> KA3 for 694.8052755548911
KA3 -> KA5 for 861.9938610958576
Average Efficiency:3.004437757217313          [+5 properly computed efficiency]
```

As a hint: *When you are done you will see that for one of these airlines, the maximal flight distance is* `4485.740310568238` *miles while the actual GPS distance is only* `433.155065627006` *miles. In this case, the efficiency is a horrible ~10.35 because the flying time is more than 10x the actual distance between the two airports.*

Version: 28-Apr-2023 11:30 PM

Q3 Search Algorithms over Graphs

On Day 21, I show how to work with highway maps as graphs. In those examples, I arbitrarily pick two vertices to conduct the search. For this assignment, you are to compute specific paths through the highway graph. Feel free to use the ideas from **`algs.days.day21.BreadthFirstPaths`** and others.

Copy the **`algs.hw4.map.MapSearch`** class into **`USERID.hw4`** and modify it to complete this question. The output is subdivided below by each question, but you should just output all at once in this class.

Q3.1 Standard Paths [10 pts.]

Using **Breadth First Search**, compute the shortest path (in terms of total number of highway segments) from the western-most highway location to the eastern most highway location. Complete the implementation of **`westernMostVertex()`** and **`easternMostVertex()`**. Be sure to identify the label associated with each vertex and print (a) the total number of edges in the path; and (b) the total distance by computing the GPS-distance of each highway segment. Your output should look like this:

```
BFS: From (SOMEPLACE-WEST) to (SOMEPLACE-EAST) has BBB edges.
BFS provides answer that is : NNN.DDDD miles.
```

Q3.2 Demonstrate why Depth First Search is inappropriate here [10 pts.]

For the same **west → east** scenario, now complete a **Depth First Search** and report the total number of edges in the computed solution, as you did with Breadth First Search. If you are unable to apply **`edu.princeton.cs.algs4.DepthFirstPaths`** consider the alternative I introduced on Day 20.

```
DFS provides answer that is : NNN.DDDDD miles with DDD total edges.
```

Q3.3 Use Dijkstra's Single Source Shortest Path Algorithm [10 pts.]

For the same **west → east** scenario, now construct an equivalent **`EdgeWeightedGraph`** and apply **Dijkstra's Single Source Shortest Path** algorithm to determine the shortest path in terms of total accumulated miles.

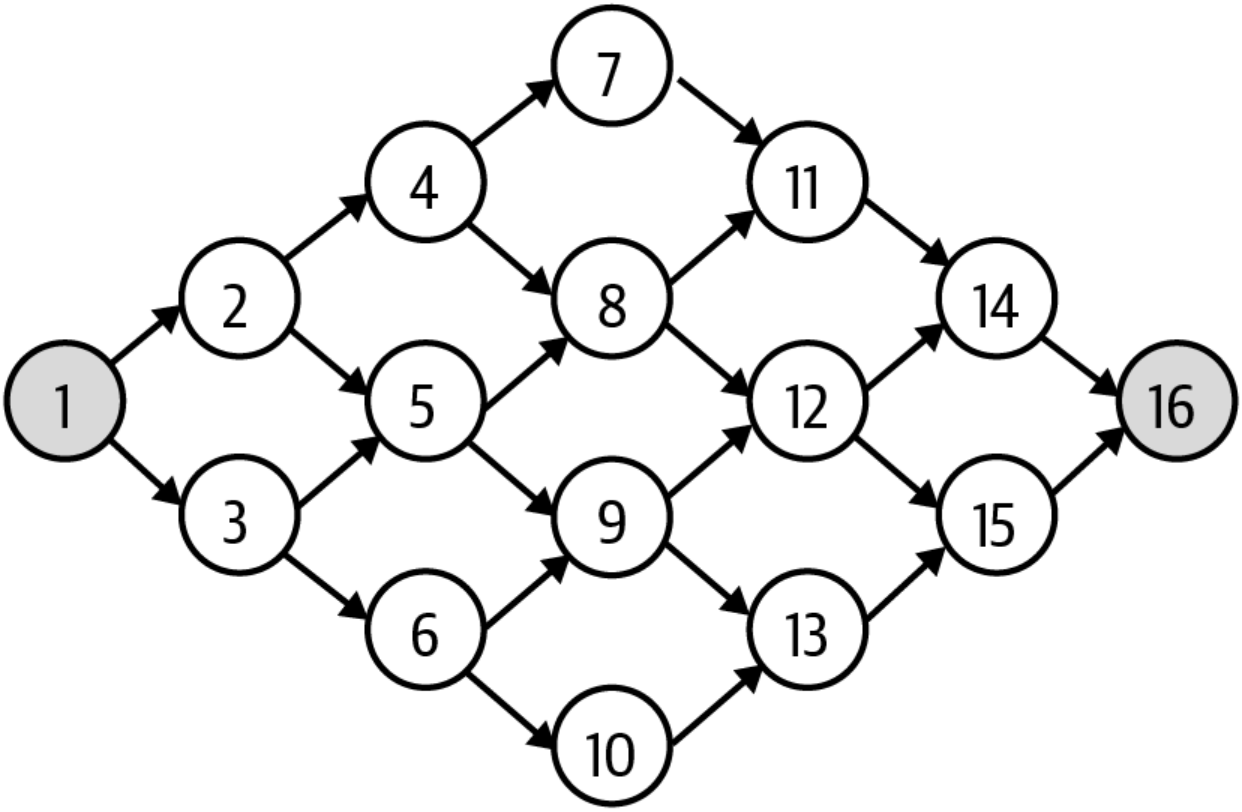
```
Shortest Distance via Dijkstra: NNN.DDDDD miles with SSS total edges.
```

*Hint: You don't have to invent the wheel! Just use the classes provided by Sedgewick (or myself) and if you do so, you will know your code works properly when **`BBB + DDD + SSS`** is equal to **`17,307`**.*

Version: 28-Apr-2023 11:30 PM

Q4 Bonus Question [1 pt.]

A directed graph with no cycles is called a *directed acyclic graph*, or DAG for short. Dijkstra's algorithm in the worst case is classified as O((E+V) log V), but for a DAG you can compute the single-source, shortest path in O(E+V). First, apply **Topological Sort** to produce a linear order of the nodes. Second, process each node, *n*, in linear order, relaxing the edges that emanate from *n*. There is no need to use a priority queue. Confirm runtime behavior on random mesh graphs where each edge has a weight of 1. In the *mesh* graph shown below the shortest distance from node 1 to node 16 is 6.

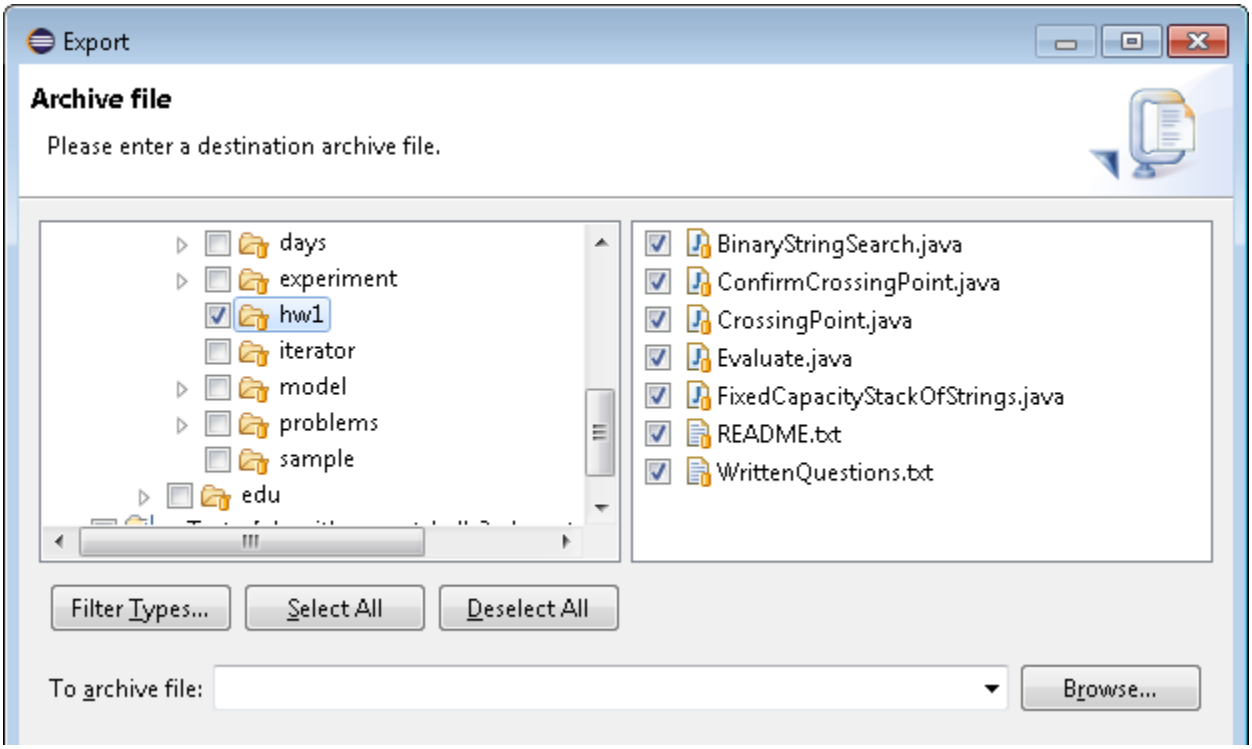


Version: 28-Apr-2023 11:30 PM

Submission Details

Each student is to submit a single ZIP file that will contain the implementations. In addition, there is a file "WrittenQuestions.txt" in which you are to complete the short answer problems on the homework.

The best way to prepare your ZIP file is to export your entire **`USERID.hw4`** package to a ZIP file using Eclipse. Select your package and then choose menu item "**Export…**" which will bring up the Export wizard. Expand the **General** folder and select **Archive File** then click **Next**.



You will see something like the above. Make sure that the entire "hw4" package is selected and all files within it will also be selected. Then click on **Browse…** to place the exported file on disk and call it USERID-HW4.zip or something like that. Then you will submit this single zip file in Canvas as your homework4 submission.

Addendum

If you discover anything materially wrong with these questions, be sure to contact the professor or TA/SAs posting to the discussion forum for HW4 on Discord;

When I make changes to the questions, I enter my changes in red colored text as shown here.

Change Notes

1. Fixed ERROR in output for Question 1 Histogram! Sorry about that
2. Clarified all of question 2 should continue to FilterLower48. I've updated accordingly.

Version: 28-Apr-2023 11:30 PM

3. Fixed values in bonus question 1.1