- Connection to 127.0.0.1:30030 was lost. Please make sure you're connected to the Internet and try again.
  (Development Only)
  details...

Connection to 127.0.0.1:30030 was lost. Please make sure you're connected to the Internet and try again.
(Development Only)
details... Close

Dashboard
SP19 COMPSCI 400 001
p1 Implement and Test DataStructureADT

Skip To Content
Dashboard

- VU PHAM Account
- Dashboard
- Calendar
- 37 unread messages 37 Inbox
- History
- Help

- *My Dashboard*
- SP19 COMPSCI 400 001
- Pages
- p1 Implement and Test DataStructureADT

Spring 2018-2019

- Home
- Modules
- Grades
-
- Assignments
- Course Syllabus (AEFIS)
- Kaltura My Media
- Kaltura Gallery
- Zoom
- Library Dashboard
- NameCoach Roster
- Course Summary

p1 Implement and Test DataStructureADT

CS 400 Individual Programming Assignment 1:
Implement and Test implementations of DataStructureADT

Announcements | Overview | Specifications | Files | Steps | Submit p1 (Links to an external site.)

Announcements

Corrections, clarifications, and other announcements regarding this programming assignment will be found below.

- **1/31: ADDED p1 Getting Started in Eclipse (Links to an external site.) (with more detail)**
  p1.zip Makefile has been updated
  - **download new Makefile (from Files section)**
  - **or change /class/ to /classes/ in each target of the Makefile you have**
- **1/28 (1:00pm): updated and added additional TA .class files for students to test**
- **1/28: class DS_My must be implemented without using Java's built-in Collection types.**
- **1/27: Program assigned.** Students must work individually on this program.
  - Use Piazza to ask questions about the assignment.
  - TAs will be available in the CS Labs starting Monday of Week 2 (See lab hours schedule)
  - REMINDER: Students caught posting coursework online (i.e. for seeking help or a paid programmer) will get a zero on the assignment for a first offense and may fail course for multiple offense. (Student Conduct (Links to an external site.))

Learning Outcomes p1 (Links to an external site.)

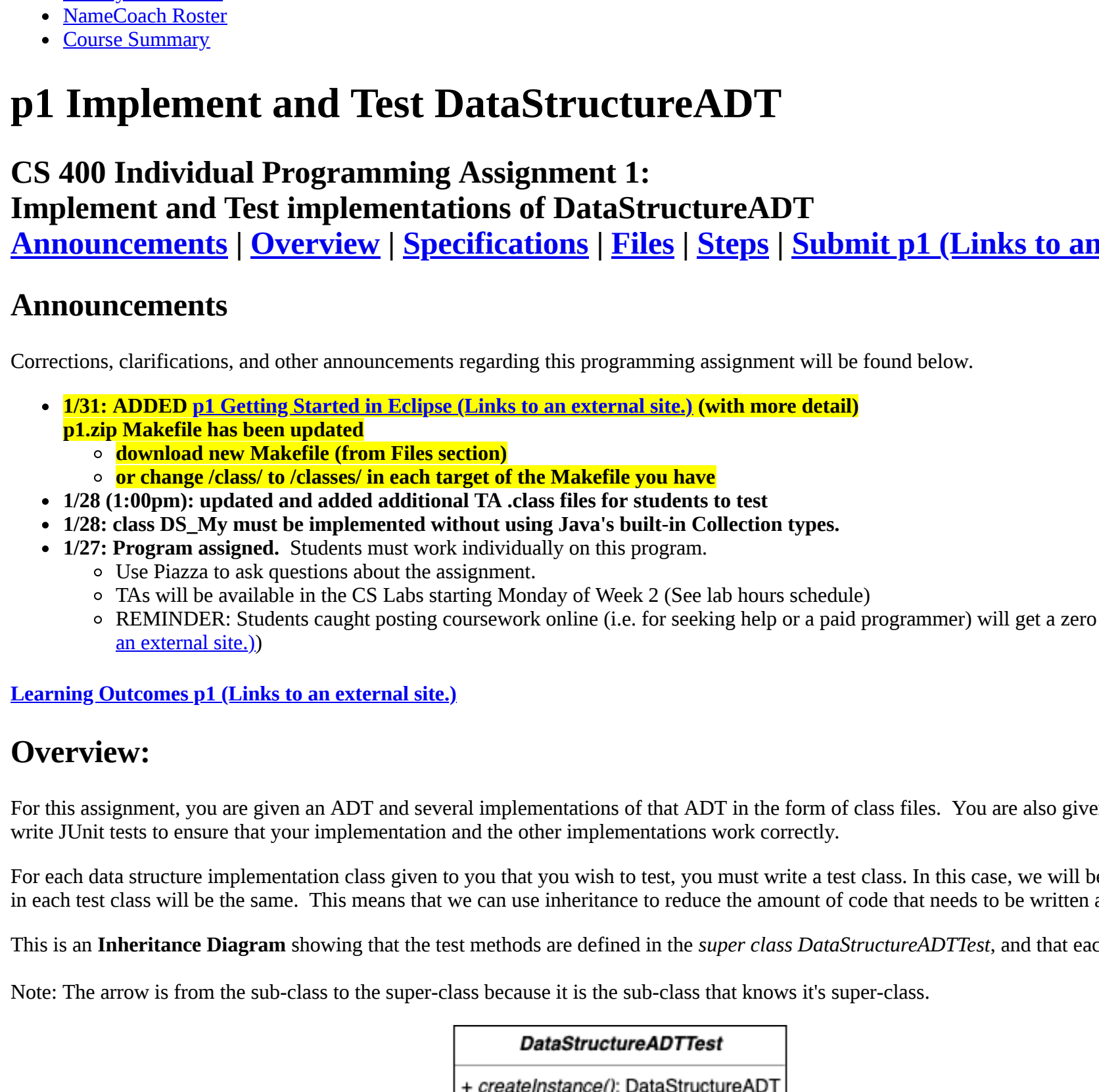
Overview:

For this assignment, you are given an ADT and several implementations of that ADT in the form of class files. You are also given an abstract JUnit super class with a few example JUnit tests. You are required to implement the ADT and write JUnit tests to ensure that your implementation and the other implementations work correctly.

For each data structure implementation class given to you that you wish to test, you must write a test class. In this case, we will be giving you many implementations to test so that means many test classes are required. However, the tests in each test class will be the same. This means that we can use inheritance to reduce the amount of code that needs to be written and still ensure that all implementations are tested with all tests.

This is an **Inheritance Diagram** showing that the test methods are defined in the *super class DataStructureADTTest*, and that each *sub-class* must only define the ***createInstance()*** method.

Note: The arrow is from the sub-class to the super-class because it is the sub-class that knows it's super-class.



Should you use Linux from the command line or use Eclipse to develop p1?

Command-line tools in a Linux terminal are great for running programs, but most command-line editors are not nearly as easy to use as the built-in editor in Eclipse and most IDEs. For that reason, many programmers develop their code in an IDE like Eclipse and use Linux for behind the scenes tasks that are not as convenient in Eclipse. For this assignment, we recommend that you create a Java Project in Eclipse and once you have developed your solution, try compiling and running it from a Linux terminal window. Students are certainly welcome to use Linux to edit and develop their code at the command-line if they prefer. Instructions for running your program on Linux are included at the end of this page.

1. Create a JUnit class in Eclipse

Using the JUnit framework means that you can focus on what tests to define and how to define them. When you run JUnit tests in Eclipse you get a graphic display showing green for passed tests and red for tests that failed. Here are some steps to create an Eclipse project that can run JUnit 5 tests. Different IDEs will likely have similar, but different instructions.

1. Create a New Java Project: **File -> New -> Java Project**
   1. **Do not use separate src and bin folders.**
2. Add a JUnit Test class: **File -> New -> JUnit Test Case**
   1. remove the package name from the wizard dialog
   2. select the "***New JUnit Jupiter Test***" option, then Finish
   3. select "***Add JUnit 5 library to Build Path***", then Finish
   4. We will not need this TestClass, so it can be deleted from the project.
3. Copy the assignment starter files to your project folder (p1.zip contains all files)
   https://pages.cs.wisc.edu/~deppeler/cs400/assignments/p1/files/ (Links to an external site.)
   1. download p1.zip (Links to an external site.) to your Eclipse project folder
   2. unzip p1.zip (you should have java files, a Makefile, and classes.zip file)
      1. If the files were placed in a sub-folder, move all files up one directory to the project folder.
   3. unzip classes.zip (if you do not have a classes folder with DS_*.class files already present)
4. Run the **TestDS_My** test class
   1. Double-click to open the **TestDS_My.java** file in the Eclipse editor.
   2. **Run -> Run As -> JUnit Test**
   3. If the above steps do not work, you could.....Go back to Eclipse, right-click on your project and choose **Refresh.......**

Note: An alternative way is to download the JUnit 5 library and place a copy of the **junit-platform-console-standalone-1.3.2.jar** in your project folder. This file is included in the p1.zip file. You will likely need to add this file to the build path: **Right-click on the jar file -> Build Path -> Add to Build Path**

2. Add Tests to the DataStructureADTTest class

Now, that your project is setup, you are ready to add tests in the **DataStructureADTTest** class. We have some required tests to help you get started. Complete the **DataStructureADTTest** class that has been started for you. This class includes comments to help guide your work at the start. But, it is not complete. There is one complete test and you must define tests 01-04 as described here. Recall: sequences that should throw exceptions must also be tested.

1. **test01_after_insert_one_size_is_one()** - insert one key value pair into the data structure and then confirm that size() is 1.
   Tip: use a call to **fail("message string")** or **assertEquals(leftsidevalue,rightsidevalue)** to get it count as a fail. If no fail occurs, it counts as a pass.
2. **test02_after_insert_one_remove_one_size_is_zero()** - insert one key,value pair and remove it, then confirm size is 0.
3. **test03_duplicate_exception_is_thrown** - insert a few key,value pairs such that one of them has the same key as an earlier one. Confirm that a RuntimeException is thrown.
4. **test04_remove_returns_false_when_key_not_present** - insert some key,value pairs, then try removing a key that was not inserted. Confirm that the return value is false.
5. TODO: add tests to ensure that you can detect any implementations that fail in some way.

Tip: consider different sequences of inserts and removes. Check results of method like: size(), contains(), remove(). Can you figure out if those methods work or not?

Run your tests on the **DS_My** class by running the **TestDS_My** JUnit 5 test class. Open the file and click the run button. Assuming that you have not yet completed your version, we expect all tests to fail.

3. Complete DS_My (your class implementation)

Implement your own data structure in the class **DS_My**. Be sure that **DS_My** implements DataStructureADT and does not have any public or package level fields, methods, or inner classes. As you implement each method in the class, run your **TestDS_My** class. You should be able see more and more tests passing as your implementation is completed. This process may also help you to identify new things to test for.

WARNING: Your DS_My may not use any Java Collection classes as internal data members. You must implement using your own arrays, linked lists, or other.

4. Create classes to test each TA implementation

Your project should have a **classes** folder that will contain the TA implementations that must also be tested. You must add this folder as an "External Class Folder" and for each TA class, make a new sub-class of the **DataStructureADTTest** class for each Test class. Don't worry if the files are GONE. Eclipse likes to do that with .class files. Just keep unzipping and re-copying them into your project as needed.

4.a Add classes as an external class folder:

1. Right-click the "**classes**" folder within your project folder
2. Select **Build-Path -> Configure Build Path**
3. Select **Libraries** tab
4. Select "**Add External Class Folder**"
5. Find the **classes** folder in the project workspace
6. **Apply and Close**

4.b Create TestDS_* classes that sub-class DataStructureADTTest

1. Right-click the **TestDS_My** class
2. Click **Ctrl-C** to copy
3. Click **Ctrl-V** to paste the copy in the same folder
4. Set the name as **TestDS_TaName** (where TaName is replaced with the particular TA implementation to be tested
5. Edit the source code in the newly created class so that it constructs an instance of the particular **DS_** type
6. Save and repeat for all other DS implementations
7. Run each TestDS_ class independently to ensure

5. Save a Screenshot of your results

After you complete your tests, make the additional Test classes, and complete your implementation in **DS_My , run All Tests** and take a screen shot of the results.

Edit the Run Configuration to run all tests at once.

- Run -> Run Configurations
- Select "Run All Tests" from the configuration dialog
- Name the configuration "Run All Tests"
- Apply and run the tests.
- Take a screenshot of your test results (they must be your results) and name it screenshot.png

Additional Test Ideas

This is an individual project and each student must figure out their own tests for this assignment. DO NOT just ask other classmates. These ideas will help you get started for things to test on your data structure.

inserts one item and fails if unable to remove it

inserts many items and fails if size is not correct

if duplicate values are not able to be inserted (and then removed) even if the duplicates are far away from each other in the sequence of inserts (hint: it is not a duplicate if one was added and later removed)

test that a key can be re-added if the key was inserted and then removed (this should not be a duplicate)

check that it can store at least 500 items, should be able remove all of them too

think of your own additional tests to ensure that you can detect problems

Files

Links are to files that can be downloaded individually.

```
 (Links to an external site.) (Links to an external site.)p1.zip (Links to an external site.) (contains all of these files and a couple of others)
DataStructureADT.java (Links to an external site.) (provided interface - DO NOT EDIT or SUBMIT)
DataStructureADTTest.java (Links to an external site.) (start of test super class is provided - YOU MUST EDIT and SUBMIT)
TestDS_My.java (Links to an external site.) (sub-class of DataStructureADTTest.java, no need to edit or handin)
DS_My.java (Links to an external site.) (start of class you must define and submit)
(classes.zip (Links to an external site.) files for each TA implementation to test)
  classes/DS_Andy.class (Links to an external site.)
  classes/DS_Deb.class (Links to an external site.)
  classes/DS_Gautham.class (Links to an external site.)
  classes/DS_Katie.class (Links to an external site.)
  classes/DS_MadanRaj.class (Links to an external site.)
  classes/DS_Niveditha.class (Links to an external site.)
  classes/DS_Roshan.class (Links to an external site.)
  classes/DS_Sapan.class (Links to an external site.)
  classes/DS_Varun.class (Links to an external site.)
  classes/DS_Vibhor.class (Links to an external site.)
  classes/DS_Yash.class (Links to an external site.)
```

Getting Started Steps

We recommended ***iterative program development*** steps for completing programming assignments.

1. **Read entire assignment**.
2. **Review the grading rubric for the assignment.**
   Note: the rubric is subject to change but it is provided to help you get a feel for what we will be looking for.
3. **Use the Java development environment of your choice.**
   These steps assume Eclipse and Java 8. You may want to review the Eclipse tutorial (Links to an external site.) to learn the basics.
   p1 Getting Started in Eclipse (Links to an external site.)
4. Eclipse, can also run all tests in the project.
   1. **Run->Run Configurations**
   2. Select option to Run All Tests
5. **Run All tests and submit ALL required files to Canvas**
6. Now, continue your incremental development efforts.
   1. Add a test that adds, gets, and removes a single item from an empty DS.
   2. **Run tests**
   3. **Submit ALL required files to Canvas**
   4. Repeat for new tests
7. Get creative. Write tests that test all functionality of your DS implementation.
8. **Run tests and submit ALL required files to Canvas**
9. Repeat "write tests - run tests" iterations until all functionality is tested.
10. **Run tests, save a screen shot in a jpg or png file, and submit ALL required files to Canvas**

Try your solution on a CS Linux Computer:

If you worked from home, you can try your p1 solution remotely on a CS computer.

Just the facts!

1. Create a project directory on the remote computer.
2. Copy p1.zip from the course web pages to your project folder, unzip p1.zip, and then unzip classes.zip
3. Copy the java files from your personal computer to the project directory on the remote computer
4. Compile and run the program on the remote computer.

The details.

1. Create a project directory in your CS account via remote computer connection

```
ssh your-cs-login@best-linux.cs.wisc.edu
mkdir -p ~/private/cs400/p1
cd ~/private/cs400/p1
cp /p/course/cs400-deppeler/public/html-s/assignments/p1/files/p1.zip .
zip -xf p1.zip              (or try unzip p1.zip)
cd p1                       (or mv p1/*.* .)
```

2. The correct directory structure should be like this:

```
├── classes
-    ├── DS_Andy.class
-    ├── DS_Deb.class
-    ├── DS_Gautham.class
-    ├── DS_Katie.class
-    ├── DS_MadanRaj.class
-    ├── DS_Niveditha.class
-    ├── DS_Roshan.class
-    ├── DS_Sapan.class
-    ├── DS_Vibhor.class
-    ├── DS_Varun.class
-    └── DS_Yash.class
├── classes.zip
├── DataStructureADT.java
├── DataStructureADTTest.java
├── DS_My.java
├── junit-platform-console-standalone-1.3.2.jar
├── Makefile
└── TestDS_My.java
```

3. **Copy your java files** from your computer to your CS project directory (overwrite existing files with same name)
   1. Open a terminal app locally
   2. Change directory to your Eclipse workspace and project directory.
      Note: actual directory depends upon your system. Use / or \ or \\ as needed)
      cd C:\users\your-account\eclipse-workspace\p1
   3. Use a secure copy program to copy all of your source code to your CS project folder (created above).
      Note: you must replace bucky with your login and /b/u/bucky/ with your user directory path
      scp *.java bucky@best-linux.cs.wisc.edu:/u/b/u/bucky/private/cs400/p1/
      or use **pscp** instead of **scp** if Windows Putty user.
4. At your remote computer terminal connection:
   1. Check the contents of the .java files and see that your **DS_My.java**, **DataStructureADTTest.java**, and other **TestDS_*.java** files are in the project on the remote computer.
   2. Compile the program: using *javac, and the junit jar file that was in p1.zip*
      javac -cp .:./classes/:junit-platform-console-standalone-1.3.2.jar *.java
   3. Run your test classes using the Java Virtual Machine (JVM): *java*
      java -jar junit-platform-console-standalone-1.3.2.jar -f TestDS_My (or any TestDS you want)
   4. Alternatively, you can use the **make** utility with the **Makefile** that was copied to your project directory.
      make all (will run all TestDS_ test classes)

Finally, don't forget to Submit all files at: p1 (Links to an external site.)