JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Syllabus
Computer Science 600.226
Data Structures
Spring, 2017 (4 credits, EQ)

Instructors

- **Section 01:**
  Associate Teaching Professor Joanne Selinski, joanne@cs.jhu.edu,
  http://cs.jhu.edu/~joanne
  Office: Malone 225
  Office hours: see website

- **Section 02:**
  Senior Lecturer Peter H. Fröhlich, phf@cs.jhu.edu, https://cs.jhu.edu/~phf/
  Office: Malone 223
  Office hours: see website

Teaching Assistant

To be announced on course website.

Meetings

- **Section 01:** Monday, Wednesday, Friday, 1:30-2:45 pm in Shaffer 301.
- **Section 02:** Monday, Wednesday, Friday, 3:00-4:15 pm in Shaffer 301.

Textbook

No textbooks are required but one is recommended and we will provide references to specific chapters/sections in the online version that correspond with course material on the main schedule:

- Clifford A. Shaffer, *Data Structures and Algorithm Analysis (Java Version)*: on-line interactive text (OpenDSA), available at
  https://opendsa.cs.vt.edu/ODSA/Books/Everything/html/
- An (older) print edition 3.2 available at
  http://people.cs.vt.edu/shaffer/Book/JAVA3elatest.pdf and through Dover Publications.

The three titles below are recommended as additional reference options. You'll likely want the most recent editions.

- Mark Allen Weiss, *Data Structures and Algorithm Analysis in Java*, Pearson Education.
- Robert Sedgewick and Kevin Wayne, *Algorithms*, Addison-Wesley. The JHU Library has some version(s) of this text in a free online format.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms,* Massachusetts Institute of Technology. This text is often required for EN.600.363/463 Introduction to Algorithms. The JHU Library has some version(s) of this text in a free online format.

For students who feel they need resources about the Java programming language beyond what is available on the Internet, the following books are suggested.

- John Dean and Raymond Dean, *Introduction to Programming with Java: A Problem Solving Approach*, McGraw-Hill. This text has recently served as the required text for EN.600.107 Introductory Programming in Java.
- Evans and Flanagan, *Java in a Nutshell*, O'Reilly. The JHU Library has some version(s) of this text in a free online format.
- Deitel and Deitel, *Java How to Program,* Prentice-Hall. The JHU Library has some version(s) of this text in a free online format.
- Arnold, Gosling, and Holmes, *The Java Programming Language*, Addison-Wesley Professional. The JHU Library has some version(s) of this text in a free online format.
- Peter Sestoft, *Java Precisely*, MIT Press.

Online Resources
The following online resources are essential:

- The course web site location at http://cs.jhu.edu/~cs226. You will find a schedule of topics, class notes, and assignment details there.
- The course Piazza site at http://piazza.com/jhu/spring2017/600226. This site will serve as our discussion site and primary communication means for the course. Please use Piazza to ask questions of the course staff and fellow students. Questions asked via email will be redirected to Piazza for answering.

Course Information

- This course covers the design and implementation of data structures including arrays, stacks, queues, linked lists, binary trees, heaps, balanced trees (e.g. 2-3 trees, AVL-trees) and graphs. Other topics include sorting, hashing, memory allocation, and garbage collection. Course work involves both written homework and Java programming assignments.

Course Goals

Upon successful completion of this course, you should be able to:

1. Evaluate and compare the time complexity of functions using mathematical techniques.

2. Design an algorithm that produces the correct results according to specified inputs and time or space complexity constraints.
3. Understand the operation of common data structures and algorithms.
4. Use analysis techniques to choose the data structure/implementation appropriate for a given problem.
5. Write advanced object-oriented solutions in Java to significant problems, by implementing appropriate data structures and algorithms.

This course will address the following ABET CAC Criterion 3 Student Outcomes

- An ability to apply knowledge of computing and mathematics appropriate to the discipline (a)
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution (b)
- An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs (c)
- An ability to function effectively on teams to accomplish a common goal (d)
- An ability to use current techniques, skills, and tools necessary for computing practice (i)
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices (j)
- An ability to apply design and development principles in the construction of software systems of varying complexity (k)

Course Topics

The goal of the course is to teach fundamental data structures, which allow one to store collections of data with fast updates and queries. Key topics will definitely include: Java refresher and generics, analysis tools, sorting, linked lists and iterators, stacks and queues, search trees, maps, hashing, priority queues, and graphs. Please see the main course website for a more detailed schedule, which will be updated as the semester progresses.

Course Expectations & Grading

Many details regarding specific course policies will be published as a separate document on the course website. See also Piazza for specific usage policies there. Below we have articulated the most essential course policies with respect to graded work.

Course grades will be based on assignments (typically Java implementations and also some written components), a midterm, and a final, according to these proportions:

- 60% - Assignments (some individual, some with others)
- 15% - Midterm (tba, in class)
- 25% - Final Exam (Sunday, May 14, 2-5p)

Each individual grade item is measured as a percentage relative to the *highest actually achieved score* for that grade item. So if the Midterm Exam was out of 40 points, but the

highest actually achieved score was 36, then whoever has 36 points gets 100% while everybody else gets a percentage *relative* to that. *If the highest actually achieved score is 0 for some reason, everybody gets 0% on that grade item. If we get the impression that you are "gaming" the relative grading system, we reserve the right to give everybody 0% on the grade item as well.* Letter grades for the course will be assigned on a standard scale, subject to the instructor's evaluation of your overall class performance. Do not expect a curve in this course.

Assignment Logistics. Blackboard will be used for all assignment submission, grades and grading feedback. Absolutely no assignments will be accepted via email or other means. These general rules apply to homework submission:

1. We will *not* accept late submissions for *full* credit. Not ever! You have to submit *before* the deadline to be sure that your submission actually arrives in time, so if you submit in the last minute you might not make it and that's your responsibility. Blackboard will not accept assignments one second beyond the stated deadlines, and neither will we.
2. We may grade (parts of) an assignment *automatically*, especially in the case of programming assignments. Following the specifications for an assignment *exactly as given* is very important for this to work. Even the "correct" output will make you lose points if you do not also follow the "correct" *format* for it. Naming your files exactly as specified is also critical in this regard.
3. Any obvious disregard for your work will get you 0 points on the assignment. For programming assignments you show obvious disregard by handing in code that either doesn't compile or that fails so miserably that we can be sure you never tested it. For written assignments you show obvious disregard by making horrendous amounts of spelling errors or handing in text files that are formatted so badly that no sane reader would want to waste their time on them.
4. Some assignments will require that you *learn something by yourself*, for example how to use a certain library that wasn't covered in detail in lecture. This is quite intentional because you have to learn how to learn something, especially in the case of programming assignments. For this and many other reasons we *strongly* encourage you to start every assignment as soon as it is posted.
5. You can submit an assignment as many times as you wish, we will grade only the last submission received before the deadline, relative to your late days and the grace period (see below). So if you are working on something with multiple parts, you can submit once after you get done with each part. That way you can be sure that we have at least a partial submission just in case you never get done with the final submission. Just make sure that *each submission contains all finished parts* as we will not fish different parts out of different submissions.
6. All parts of an assignment must be submitted together in a complete zip file, with the required folder structure and file names. Do not include extraneous files or folders. You should always check your submission by downloading it from Blackboard, opening it in a fresh folder and reviewing it (compiling, running, testing, etc.) as if you were grading it yourself. If anything is missing, you can then make an adjustment, repackage into a complete zip, and resubmit before the deadline (but only if you allow yourself enough time to do this!). If we discover

while grading that necessary parts of an assignment are missing, it will almost certainly be too late to resubmit them and your grade will suffer the consequences.

The implementation projects in this course will require you to design and write Java programs that compile with the standard Java 8 compiler and run on the virtual machine that we distribute. You will receive no credit for programs that do not compile on this virtual machine. In addition, students may request a Linux account from the Computer Science Department, and access to the CS Undergraduate Lab in Malone Hall 122 to work on assignments and meet with course assistants.

You are also free to download a Java compiler and do your work on your own computer, though it is recommended that you use either a Unix/Linux environment or an integrated development environment such as Eclipse for your work in this course. If working on your own machine, you should always check that your code compiles and runs correctly on the virtual machine since assignments will be graded there. It is your responsibility to ensure that all the course tools are installed correctly on your machine; incorrect installations or use are likely to result in grading deductions.

Attendance. All students are generally expected to attend all meetings of this course, and actively participate in all course meetings. If you miss a class meeting for any reason, you are responsible for material presented, and it is your responsibility to obtain any missed handouts or other materials. (See also Illness and Religious Holidays.)

Late Policy. We will have a grace period, and a late days policy for this course. Here's how they each will work, and the combined effect:

Grace Period: Assignments will have a grace period of *one hour* after the initial deadline that incurs a *10% grade penalty*. If you submit within the hour following the stated deadline, your submission will be accepted and no late days will be assessed, but your total earned score for the assignment will be reduced by 10%. For example, if you earn an 85 on an assignment, but it was submitted in the hour of grace, then the resulting grade will automatically be reduced by 10% to a 76.5. The grace period only applies to the initial submission deadline, not to assignments submitted with late days. Therefore (see below), no assignments can be submitted or will be accepted by more than 48 hours after the initial deadline.

Late Days: All students start the semester with a budget of *five late days*; however, you can only use *at most 2 at a time* for any one submission deadline. A late day is defined as exactly 24 hours, and does not come with a grade penalty. Late days are assessed as whole days only, no prorating. You can only use a late day if you have one left in your budget; a pair or team can only use one if *all* members have a day left. Late assignments will not be accepted if your late days have been used up, meaning that you will get a 0 if nothing is submitted by *your* deadline, relative to *your* late days. If you submit in the first hour grace period, you will automatically be assessed the 10% grace penalty, not a late day. If you prefer to use a late day than get penalized, you must wait until the hour of grace has passed to submit your assignment

(or resubmit after that first hour). If your final submission is later than your remaining late days allow, it will be ignored and we will grade the last submission that is within *your* late days budget, if one exists. Otherwise, you will get a 0. You do not have to ask permission or notify us if using a late day or two; we will automatically grade your submissions according to these policies.

Example: We will determine which submission to grade and penalties to assess based on when you submit as follows. Suppose a stated assignment deadline is 10pm on Feb 5. If your final submission is *before* 10pm on Feb 5, you will not have any late or grace penalties. If your final submission is between 10pm and 11pm on Feb 5, you will receive the 10% grace penalty, but not lose any late days. If your final submission is between 11pm on Feb 5 and 10pm on Feb 6, and you have at least one late day left in your budget, that assignment will be graded without penalty and you will be charged a late day. If your final submission is between 10pm on Feb 6 and 10pm on Feb 7 and you have >=2 late days remaining, that assignment will be graded without penalty and you will be charged 2 late days. Submission will not be possible for anyone after 10pm on Feb 7, not even one second past that deadline. Remember, if your final submission is later than your remaining late days allow, it will be ignored and we will grade the last submission that is within your late days budget. Take advantage of multiple submissions so that we always have something to grade for you within the deadlines. We will be very strict about these deadlines and policies.

Late days are a valuable commodity and you should use them sparingly. They are primarily intended to help you deal with unexpected circumstances, such as illness and technical problems, so you should not make them part of your regular planning. (Also see Illness and Religious Holidays.)

Technical Disasters. You are responsible for keeping your work safe and in a state that is ready for submission regardless of the glitches that life throws your way. These glitches include hard-disk crashes, accidentally deleting your source code, and dogs eating your USB sticks for example. Luckily there are many ways to safeguard your work, including regular back-ups, use of dropbox, version control systems, storing files on the ugrad server (backed up nightly), etc. Assignment deadlines will not be extended for these types of problems – avoid them, be prepared for recovery, and use late days as needed.

Key Dates, Assignments & Readings
Assignments will be distributed on the course webpage and due almost every week.
Midterm: tba, Final Exam: Sunday, May 14, 2-5p.

Illness and Religious Holidays
If you miss class now and then due to a religious holiday or minor illness, you do not need to inform us. Assignment deadlines will not be extended for these situations – instead use your late days if accommodations are needed.

If you have to miss significant class time or an exam, or need further assignment accommodations due to prolonged illness or religious holidays, please inform us as soon

as possible to discuss the details. In case of a long-term illness however, you must request a doctor's note and give it to us once you return to school; details of the illness are not important, but a clear statement that you could not attend school for a certain amount of time is required. Note that Student Health and Wellness does not provide written doctor's notes, but you can have them contact the Dean of Student Life who in turn will contact your instructors, including us.

Disability Accommodations
Any student with a disability who may need accommodations in this class must obtain an accommodation letter from Student Disability Services, 385 Garland, (410) 516-4720, studentdisabilityservices@jhu.edu. Present that letter to your section instructor, and also remind us several weeks in advance of each exam to make accommodations.

Ethics
The strength of the university depends on academic and personal integrity. In this course, you must be honest and truthful, abiding by the *Computer Science Academic Integrity Policy*:

> Cheating is wrong. Cheating hurts our community by undermining academic integrity, creating mistrust, and fostering unfair competition. The university will punish cheaters with failure on an assignment, failure in a course, permanent transcript notation, suspension, and/or expulsion. Offenses may be reported to medical, law or other professional or graduate schools when a cheater applies.
>
> Violations can include cheating on exams, plagiarism, reuse of assignments without permission, improper use of the Internet and electronic devices, unauthorized collaboration, alteration of graded assignments, forgery and falsification, lying, facilitating academic dishonesty, and unfair competition. Ignorance of these rules is not an excuse.
>
> Academic honesty is required in all work you submit to be graded. Except where the instructor specifies group work, you must solve all homework and programming assignments without the help of others. For example, you must not look at anyone else's solutions (including program code) to your homework problems. However, you may discuss assignment specifications (not solutions) with others to be sure you understand what is required by the assignment.
>
> *If* your instructor permits using fragments of source code from outside sources, such as your textbook or on-line resources, you must properly cite the source. Not citing it constitutes plagiarism. Similarly, your group projects must list everyone who participated.
>
> Falsifying program output or results is prohibited.
>
> Your instructor is free to override parts of this policy for particular assignments. To protect yourself: (1) Ask the instructor if you are not sure what is permissible. (2) Seek help from the instructor, TA or CAs, as you are always encouraged to do, rather than from other students. (3) Cite any questionable sources of help you may have received.

On every exam, you will sign the following pledge: "I agree to complete this exam without unauthorized assistance from any person, materials or device. [Signed and dated]". Your course instructors will let you know where to find copies of old exams, if they are available.

The simplest rules to follow are: *do all your own work* and *cite all sources*. When in doubt about the ethics of an action, probably it is not permitted but you should ask to be sure. In addition, the specific ethics guidelines for this course are:

1. In the completion of homework assignments, you may not discuss your approach with or show specifics of your code to others, unless working together in a sanctioned pair or team. This includes fellow students, former students, friends, etc. You are permitted to request assistance from course staff (instructors, TAs and CAs) only.
2. You are permitted and expected to reuse and adapt code from lectures and the assigned text (Shaffer - either version) in completing your projects. However, all original sources must be cited in comments within your code.
3. In using Piazza to ask questions about homework assignments, you must post *privately* to *all Instructors* any questions that involve significant code or that would give away your approach to solving the assignment. Otherwise, you are encouraged to publicly ask general, abstract questions, or specific questions with different code examples, so other students may benefit from the discussion.
4. You must not maintain your code in a public repository, or on a public machine, or otherwise make it available to others (intentionally or unintentionally).

Report any violations you witness to the instructor.

You can find more information about university misconduct policies on the web at these sites:

- For undergraduates: http://e-catalog.jhu.edu/undergrad-students/student-life-policies/
- For graduate students: http://e-catalog.jhu.edu/grad-students/graduate-specific-policies/