# CS5004-5005-SV – Lab 12 – July 23, 2020
# *Collections*

This week's lab provides an opportunity to explore and gain practice with Java's ***Collections<T>*** interface, applying it to a game and puzzle. There are four concrete classes of interest: TreeSet<T>, HashSet<T>, ArrayList<T>, and LinkedList<T>. (As mentioned in lecture, Vector<T> is not generally used in new code.) Tonight, we focus on HashSet and LinkedList. Start by pulling up the lecture slide showing The Collection Landscape and grabbing a *copy* of **HashDemo.java** from the lecture, for reference.

0. ***Requests***. Last week, you were asked to provide suggestions for topics or activities that might be helpful in Lab. A summary of requests so far has been provided as a handout (courtesy of our TA, Nancy!). The top item listed was "interfaces." Today's lab relies on generic interfaces and abstract classes in the space of Collections, so there will be practice using those interfaces. Let's begin, though, with open-ended Q&A about interfaces. (Nothing to turn in for this item.) Next Lab may have something more specific to work on, in the way of practicing with interfaces.

1. Create a PlayingCard class. There are 52 unique playing cards. Each card has a suit and a rank. You may use Strings to represent the four suits (e.g., "Hearts"), and the integers 1 through 13 to represent the rank (Ace = 1, Jack = 11, etc.). Remember that when you create a new class, you should create suitable constructors, getters, and setters. You should also override .toString, .equals, and .hashCode. Question to answer in a comment: How big should the hash table be to balance good performance versus size, given that there are exactly 52 unique PlayingCards?

2. Use your PlayingCard class to implement a basic "Blackjack" (a.k.a. "21") card game. See: https://planningwithkids.com/2012/06/20/21-card-game/. You will probably want to use a HashSet<T> to represent the **deck** and the cards in each player's **hand**. Just implement a 2-player version of the game (that is, the computer deals and two players each try to get 21, alternating turns without "going bust." Methods should include drawing a card at random from the deck, determining if a hand has lost (and declaring the other player the winner), printing the current visible game state on the

console, and asking player's whether to "hold" or "hit." You may want to use a boolean variable for whether or not to show the "hole card" (which is normally face-down in a real game) to aid in debugging. Use the built-in methods of the Set<T> interface to implement your game. This URL may help address the question of how to choose random elements from an (unordered) HashSet: https://www.javacodeexamples.com/get-random-elements-from-java-hashset-example/2765

***3.*** Use a derived class from *Collections* to represent each of 3 towers in the famous puzzle, ***Towers of Hanoi:*** *https://www.mathsisfun.com/games/towerofhanoi.html.* *Each of the rings in the class may be represented by an Integer, but* ***order matters*** *and you* ***may not want to assume that rings are unique****.* It might be useful to define Tower as extending LinkedList<Integer>. Try to primarily use methods that are part of the List<T> interface; but you will need to also implement a **move** method to move the top ring from one tower to another, e.g., public boolean towerA.move(Tower towerB). Return true if the move was successful. Your move method should update the contents of each tower, using add and remove, and print a description of the move that was made to the console. Remember that you must not put a larger ring on top of a smaller one. HINT: Recursion may be helpful in solving this puzzle.

**4.** [OPTIONAL] A simple text-based graphical representation of the state of the three towers might be nice.

NB: you can probably find many solutions to this by searching the web. Try to think it through on your own, first, please!

***Submit your answers to above items to Canvas for CS5005*** *(due next Thursday by midnight). Only your .java files and any associated .txt files for the above need to be uploaded. (Often you can just add comments within your code files as needed to identify the problems and to answer questions about the code or clarify your approach.) Timely submission encouraged, but late is better than never. Then work on any unfinished labs or CS5004 homework assignments. Remember to submit* ***homeworks to Canvas-for-CS5004*** *and* ***labs to Canvas-for-CS5005****.*