Array practice

- Write and test a function that takes an array of doubles and returns the average of the values in the array
- Write and test a function that takes an array of doubles and returns number of values in the array that are above the average of the values in the array
- Write and test a function that takes an array of Strings and returns number of values in the array that start with an uppercase letter. Some builtin functions you may use:
  String Class:
  ```
  char charAt(int index)
  Returns the char value at the specified index.
  ```
  Character Class:
  ```
  static boolean isUpperCase(char ch)
  Determines if the specified character is an uppercase character
  ```

Objects/Arrays of Objects

- Create a Class that represents a date with a month and a year.
- Create a Class that represents a Trip with an origin, a destination, a duration in hours, a mode of transport and a date of travel (leverage the Date class you just designed.
- Create an array of Trips that you can use to test the following methods:
  - Write a method that takes an array of trips and returns a list of all of the destinations in the list of trips.
  - Write a method that takes an array of trips and returns the Trip with the longest duration.
    - Additional: have the method throw an appropriate exception if the array of trips is empty

List ADT

- Write/test the following functions in a LinkedList and ArrayList that hold integers
  - A function that returns the sum of all values in the list
  - Write and test a function that takes an list of doubles and returns the average of the values in the list
  - Write and test a function that takes an list of doubles and returns number of values in the list that are above the average of the values in the list
  - Write and test a function that takes an list of Strings and returns number of values in the list that start with an uppercase letter. Some builtin functions you may use:
    ```
    String Class:
    char charAt(int index)
    Returns the char value at the specified index.
    Character Class:
    static boolean isUpperCase(char ch)
    Determines if the specified character is an uppercase
    character
    ```
  - A function that returns true if all numbers in the list are above a given threshold and false otherwise
  - A function that negates all of the values in the list (that is multiplies each element by -1)
  - Traverse the list and keeps only positive elements (> 0)
  - A function that takes another list as a parameter and determines whether the lists are equal or not. To be equal they must be of equal lengths and have the same elements in the same order.

- A function that takes another list as a parameter and appends it to the end of the existing list .
- A function that will insert a given value into the current list assuming the current list is sorted in increasing order.
- A function that takes another list as a parameter and interleaves the values of each list. The first element in the existing list should remain the first list. For example, if l={1, 4, 5} and the other list={7, 2, 3} should result in l={1,7,4,2,5,3}.
  If one of the lists is larger than the other, the function should append the remaining elements to the end.
  For example, if l={1, 4, 5} and the other list={7} should result in l={1,7,4,5}.
  Or, if l={1,5} and the other list={7, 6, 9} should result in l={1,7,5,6,9}.

Stack ADT

- Write a function that takes a string, and uses a stack of characters to create a reversed version of the string and return it. Example, if the function is called with "abcd" it should return "dcba" and it must use a stack to solve the problem.

Binary Trees

- Write/test the following functions in a BinaryTree class with integer values
  - A function that returns the height of this tree
  - A function that determines whether a tree is full or not.
    A full tree a tree is a tree in which the bottom level has no null trees – that is, the second from bottom level nodes will all have non-null left and right subtrees
  - A function that returns the biggest value in the tree.
  - A function that returns the sum of all of the values in the tree.
  - A function that prints an inorder traversal of the tree
  - A function that prints a preorder traversal of the tree
  - A function that prints a postorder traversal of the tree
  - A function that prints a level order traversal of the tree

- Write/test the following functions in a BinarySearchTree class with keys that are Strings and values that are Integers
  - A function that searches the tree for a given key and returns the value if the key is found. You should throw a key not found exception if it is not found.
  - A function that searches the tree for a given value and doubles the value if the key is found and does nothing if it is not found.

Hashing

- Using the built in ArrayList<T> provided by Java, create your own HashMap class with separate chaining collision resolution.
  You cannot create a generic array ie. You cannot say: `T[] array = new T[100]`
- Using the built in ArrayList<T> provided by Java, create your own HashMap class with linear probing collision resolution. Again, you cannot create a generic array.

Inheritance

- Use lecture files to review behaviour