Exercise Set 5: Trees

Exercise 5.1 A river is defined by the cities at the places where two tributaries join together, or at the river source. If we want to study river traffic or river pollution, we may select the following representation of the river:

`City` is a structure, which consists of

- `name`
- `location` represented as double x-y coordinates
- `data` - an object in the class `Data`, which we can define as we wish, to study various problems related to this river, such as pollution, fish population, navigability

`ARiver` is

- either a `Port`, which contains just one member data field - a `City`
- or a `Tributary`

`Tributary` is a structure, which consists of

- `Port`
- `leftTributary`
- `rightTributary`

For this assignments perform the following tasks:

1. Design the class hierarchy corresponding to the above data definition. Make class *stubs* with data definitions, constructors, and `toString()` methods. Make sure you test these.
2. Draw UML diagram for this class hierarchy
3. Test your code by drawing the river using the given `RiverDisplay` class.
4. Write the method `navLength()`, which computes the total length of all tributaries for this river.
5. Write the method `findCity()`, which determines whether the city with the given name is a port on this river.
6. Write the method `riverLength()`, which determines the length of this river (that means the longest distance from the source to the river end).
7. *Optional*: Add a field to measure the volume of pollution that flows into the river from each city. Define the formula to measure the pollution in the river, which is based on the length of the river segment and the total volume of pollutants. Write the method to compute the pollution for the river.

Exercise 5.2 A town soccer team has a phone tree to notify players about game cancellations and other changes in the schedule. Each player is assigned to call at most two other players.

- Develop the classes to represent the phone tree, where the player information consists of player's name and phone number.
- Develop the method `countPlayers`, which counts how many players are in the phone tree.
- Develop the method `isInTree`, which determines whether a player with the given name is in this tree.
- Develop the method `phoneNumber`, which returns the phone number of a given player, or returns 0 if the player is not found in the tree.
- Develop the method `myPhoneTree`, which returns the phone subtree starting with the given player. If the player is not found, it should return the empty tree.
- Develop the method `checkList`, which determines whether every player in a given list of players also appears in the given phone tree.

Exercise 5.3 Design the `class Item`that contains some integer-valued *attribute* that defines an ordering of Item objects. For example, an item may be a book (with price), a CD (with number of tracks), a circle (with area), an animal (with weight)

- Develop the classes to represent a binary search tree of `Item`s.
- Develop the method `insert`, which inserts a new `Item` into the BST.
- Develop the method `inBST`, which determines whether an `Item` with the given *attribute* appears in the BST.
- Develop the method `find`, which produces the `ConsBST`, which has the `Item` with the given *attribute* as its root, or it produces an `EmptyBST` object if the `Item` is not found.