

Assignment 6

Objectives

- Introduction to the Binary Search Tree ADT
- More practice with generics
- Introduction to tree traversals
- Introduction to the Map ADT.

Introduction

In this assignment you will implement a binary search tree for which each node will contain both a key and a value associated with that key. The elements in the tree will be ordered by the key stored at each node. After you have completed implementing the binary search tree, we will use that implementation to implement the Map ADT.

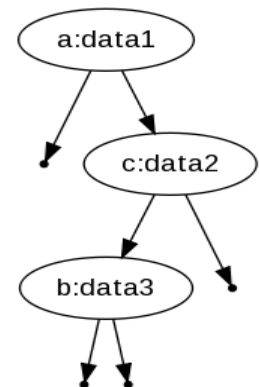
Quick Start

1. Read this entire document first once through!
2. Read the comments in `BinarySearchTree.java` carefully and implement the methods until all the tree tests pass in `A6Tester.java`
Debugging tips are provided in Part I on the following page.
3. Read `Map.java` carefully and implement the methods in `BSTMap.java` until all the map tests pass in `A6Tester.java`

Part I - BinarySearchTree

Your binary search tree implementation will use generics to allow users of the tree to specify the type for both the key and the value. For example, the following code produces the tree shown:

```
BinarySearchTree<String, String>    t1;  
t1 = new BinarySearchTree<String,String>();  
t1.insert("a", "data1");  
t1.insert("c", "data2");  
t1.insert("b", "data3");
```



When testing your code, you may find it useful to look at the trees the created and used by `A6Tester.java` (A visual representation of each of the four test trees is provided on the following pages of this assignment description.)

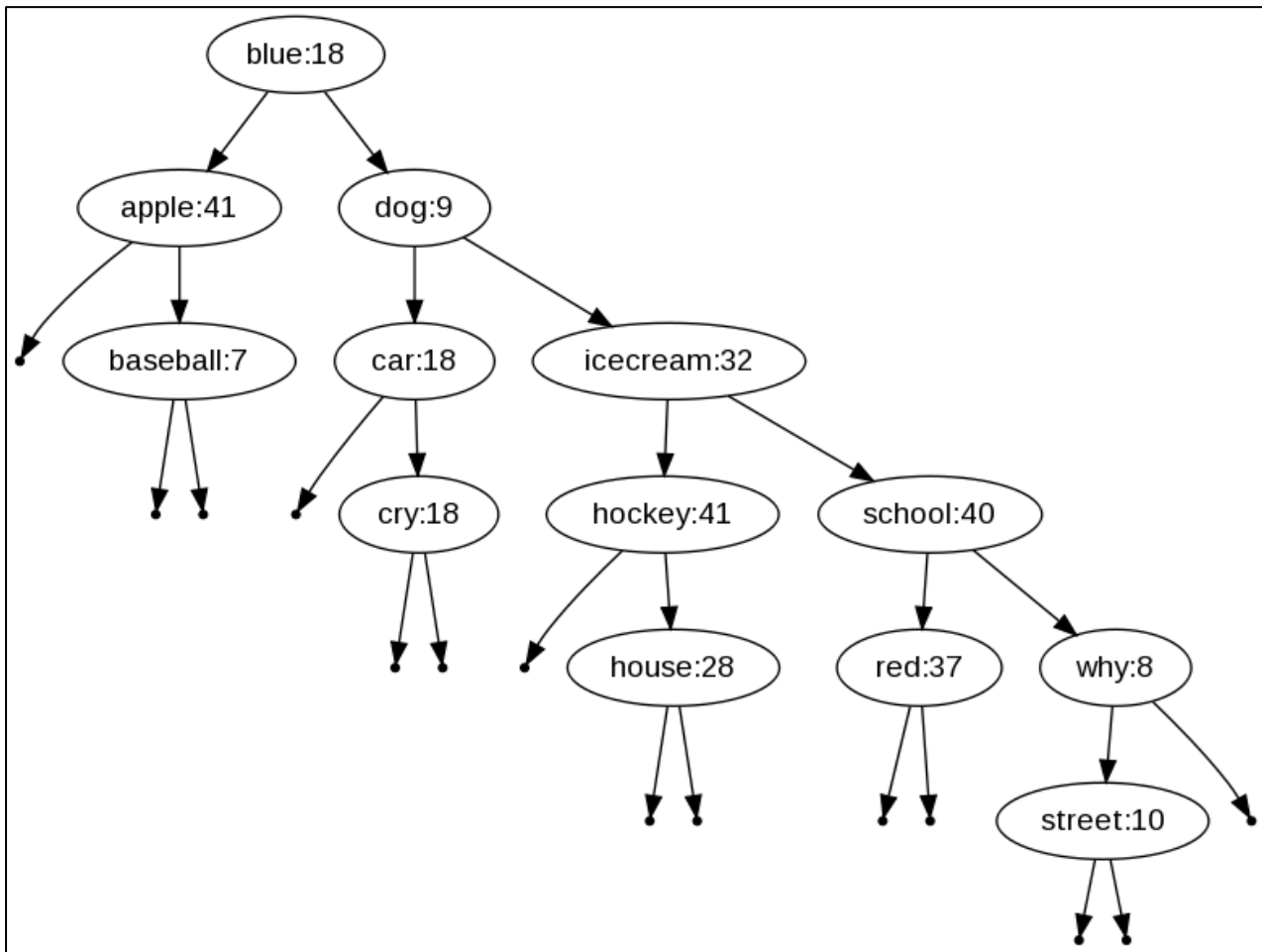
Since it is helpful to be able to visualize trees, you've been provided with a `TreeView` class which takes an instance of a binary search tree as the constructor's parameter. For example, if after the code shown above you use the following code:

```
TreeView<String,String> v1 = new TreeView(t1);  
v1.dotPrint();
```

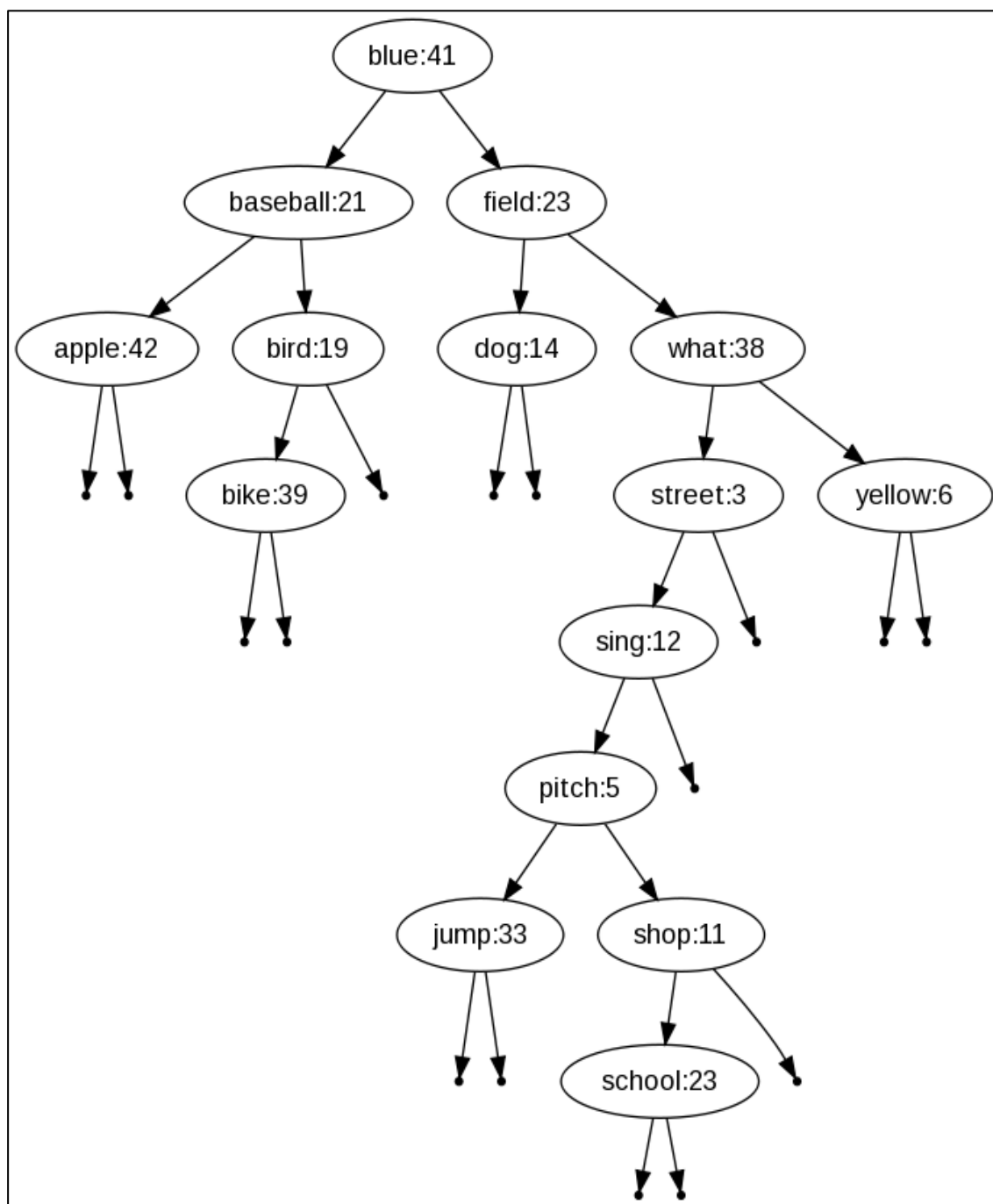
The result will be the contents of a dot file printed to the standard output. You can copy and paste that output text into the following web page to get a picture of your tree:

<http://sandbox.kidstrythisathome.com/erdos/>

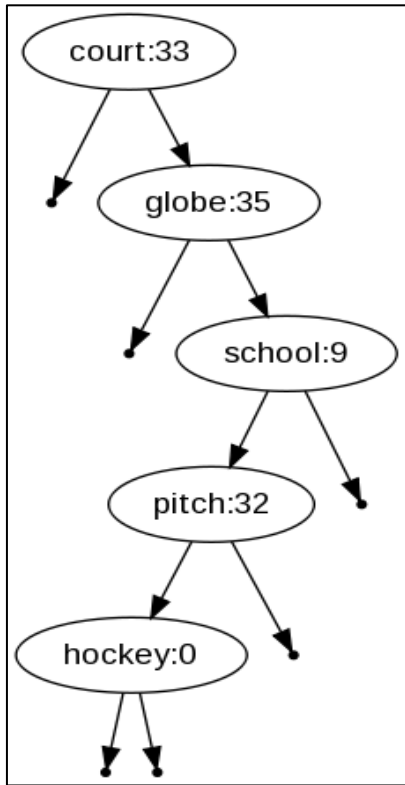
The following images are visual representations of the 4 trees generated using the TreeView class as described above in A6Tester.java



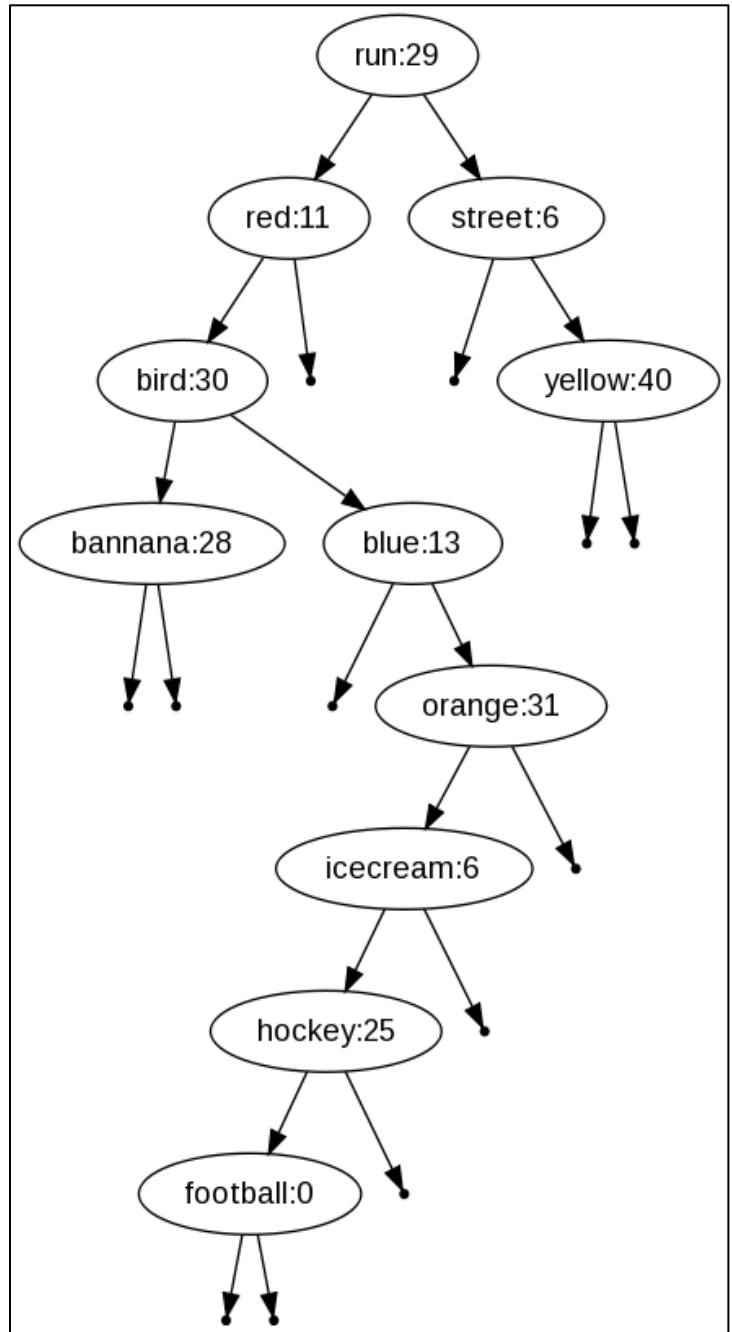
Tree 1



Tree 2



Tree3



Tree 4

Part II – BSTMap

Once you've finished implementing the Binary Search Tree, you will use this to implement the Map ADT in the file `BSTMap.java`. The Map interface is specified in `Map.java`, and provides two main operations:

- `get(key)` – return the value stored at key
- `put(key,value)` – insert the key/value pair into the map. If the key already exists, update the value stored at key to be the new value.

You have to write very little code to accomplish this – simply create an instance of your `BinarySearchTree` in the `BSTMap` constructor and then use methods you have already completed in the `BinarySearchTree` to provide the services required by the Map¹.

(In your instructor's sample solution, all methods except `containsKey` are only one line. `containsKey` contains seven lines.)

Submission

Submit only your `BinarySearchTree.java` and `BSTMap.java` via `conneX`.

Please be sure you submit your assignment, not just save a draft. ALL late and incorrect submissions will be given a ZERO grade.

If you submit files that do not compile, or that do not use the correct method names you will receive a **zero grade** for the assignment. It is your responsibility to ensure you follow the specification and submit the correct files.

Your code must **not** be written to specifically pass the test cases in the testers, instead, it must work on all valid inputs. We will change the input values, add extra tests and we will inspect your code for hard-coded solutions.

A reminder that it is OK to talk about your assignment with your classmates, and you are encouraged to design solutions together, but each student must implement their own solution. We will use plagiarism detection software on your assignment submissions.

¹This is a common practice. You can read about it here:
https://en.wikipedia.org/wiki/Adapter_pattern