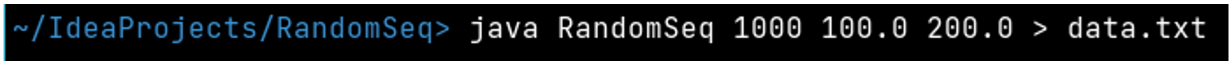
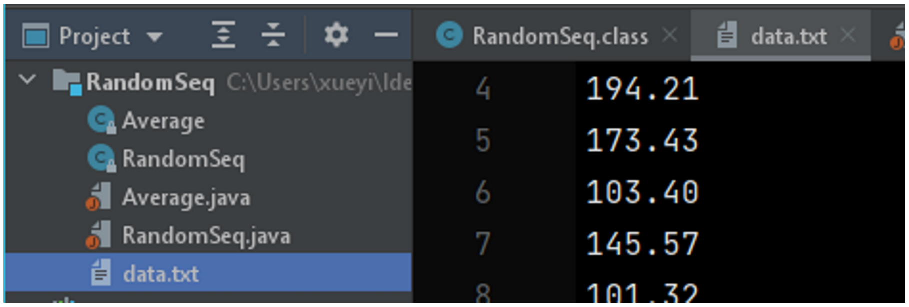
Redirection and Piping

Monday, April 11, 2022 8:58 PM

- Storing the standard output stream of a program to a txt file, instead of printing out on the terminal window

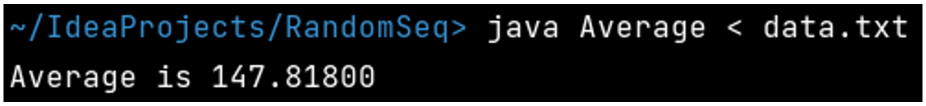


```
~/IdeaProjects/RandomSeq> java RandomSeq 1000 100.0 200.0 > data.txt
```



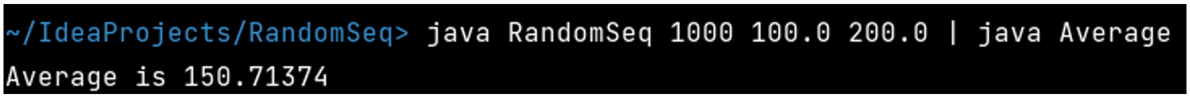
```
4    194.21
5    173.43
6    103.40
7    145.57
8    101.32
```

- Using txt file as the standard input of a java program



```
~/IdeaProjects/RandomSeq> java Average < data.txt
Average is 147.81800
```

- Combining the above actions to **redirect the output of one program to the input of another** is known as ***piping***

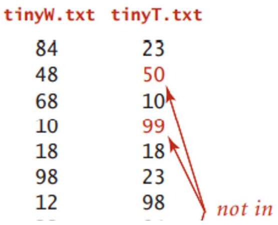


```
~/IdeaProjects/RandomSeq> java RandomSeq 1000 100.0 200.0 | java Average
Average is 150.71374
```

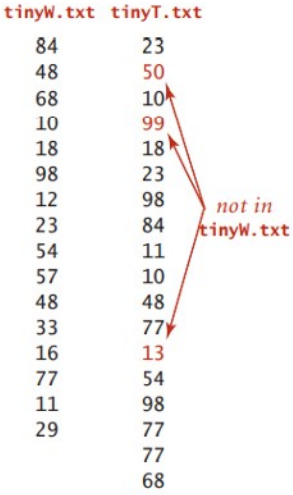
This command specifies that standard output for RandomSeq and standard input for Average are the same **stream**

A **stream** is a continuous flow of data(which are really arrays) that you don't necessarily know where the data is coming from. Stream can be an abstraction that describe for files, terminal input/output, etc.

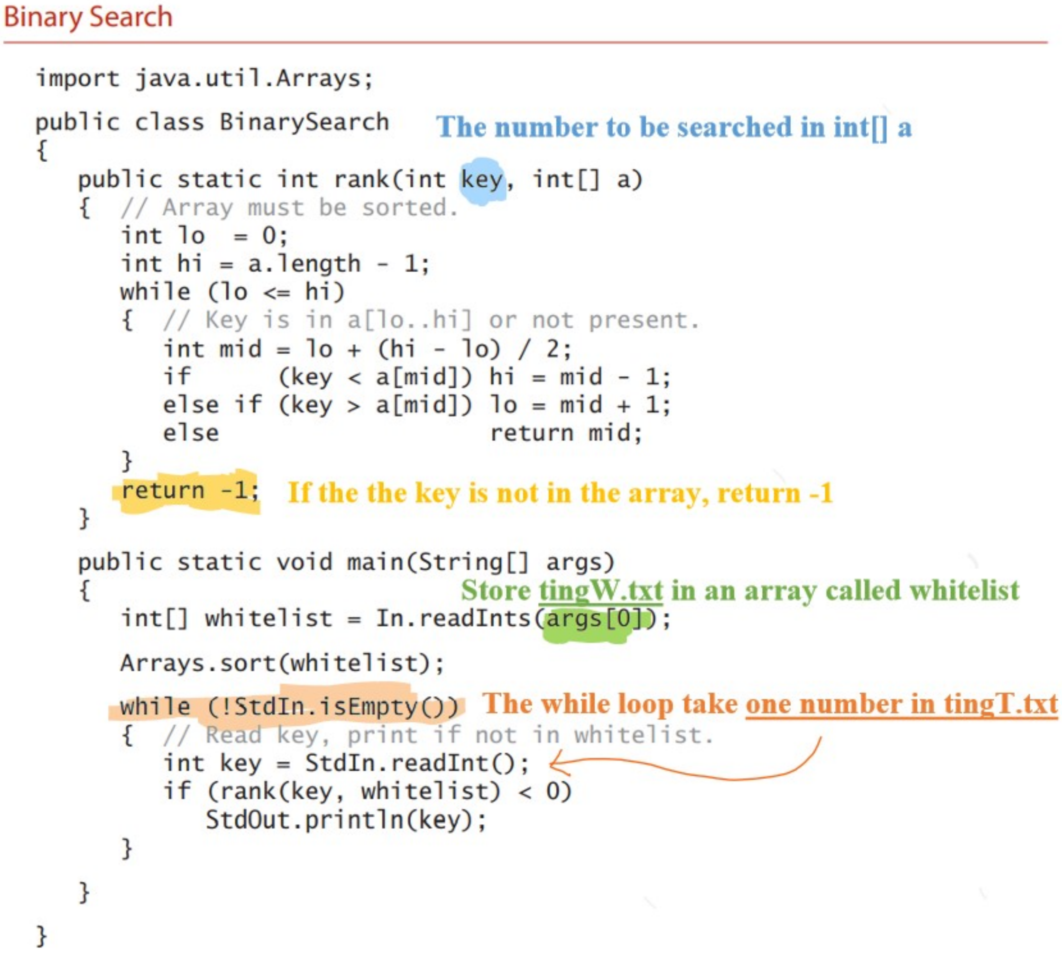
- Piping with binary search program



```
tinyW.txt  tinyT.txt

   84         23
   48         50
   68         10
   10         99
   18         18
   98         23
   12         98
```



```
tinyW.txt  tinyT.txt
  84         23
  48         50
  68         10
  10         99
  18         18
  98         23
  12         98
  23         84
  54         11
  57         10
  48         48
  33         77
  16         13
  77         54
  11         98
  29         77
             77
             68
```



Binary Search

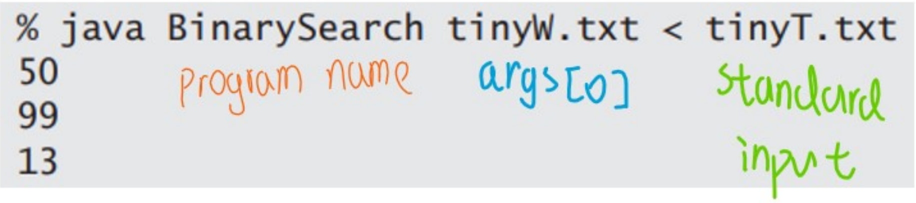
```
import java.util.Arrays;

public class BinarySearch
{
   public static int rank(int key, int[] a)
   {  // Array must be sorted.
      int lo  = 0;
      int hi = a.length - 1;
      while (lo <= hi)
      {  // Key is in a[lo..hi] or not present.
         int mid = lo + (hi - lo) / 2;
         if      (key < a[mid]) hi = mid - 1;
         else if (key > a[mid]) lo = mid + 1;
         else                   return mid;
      }
      return -1;
   }

   public static void main(String[] args)
   {
      int[] whitelist = In.readInts(args[0]);

      Arrays.sort(whitelist);

      while (!StdIn.isEmpty())
      {  // Read key, print if not in whitelist.
         int key = StdIn.readInt();
         if (rank(key, whitelist) < 0)
            StdOut.println(key);
      }

   }
}
```

The number to be searched in int[] a

If the the key is not in the array, return -1

Store tingW.txt in an array called whitelist

The while loop take one number in tingT.txt for each iteration



```
% java BinarySearch tinyW.txt < tinyT.txt
50
99
13
```

Program name

args[0]

standard input