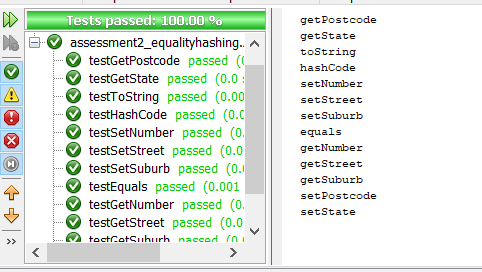
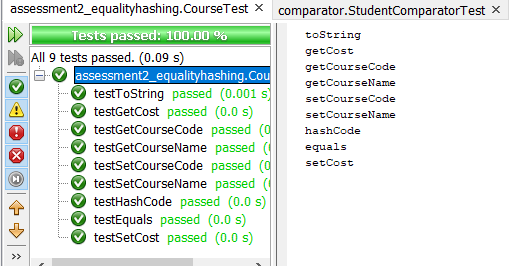
# Assignment Documentation

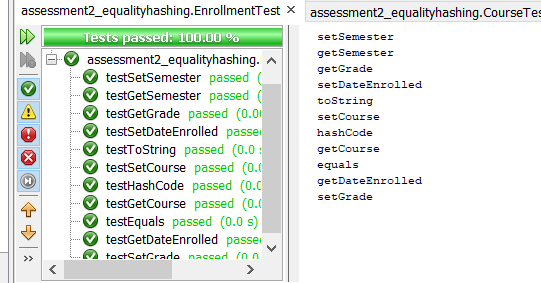
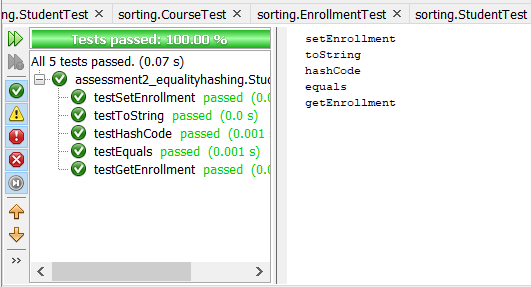
## Equality and Hashing Assignment

Deskcheck

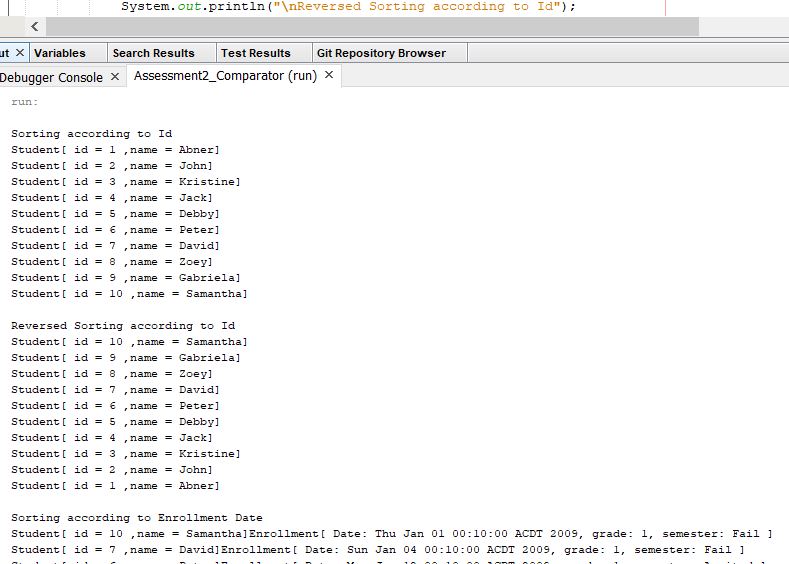
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Person1.Name | Person1.Email | Person1.TelNum | Person1.Address | Person2.Name | Person2.Email | Person2.TelNum | Person2.Address | Result expected | Actual Result |
| P1 | James | James@yahoo.com | 04536789 | 37, leader street, Goodwood.  SA-5034 |  |  |  |  |  |  |
| P2 |  |  |  |  | John | John.Smith@yahoo.com | 045454578 | 65, Pirie Street, Adelaide.  SA-5000 |  |  |
| P1.GetHashCode() == p2.GetHashCode() |  |  |  |  |  |  |  |  | false | false |
| P1 | James | James@yahoo.com | 04536789 | 37, leader street, Goodwood.  SA-5034 |  |  |  |  |  |  |
| P3 |  |  |  |  | James | James@yahoo.com | 04536789 | 37, leader street, Goodwood.  SA-5034 |  |  |
| P1.GetHashCode() == p3.GetHashCode() |  |  |  |  |  |  |  |  | true | true |
| P4 | james | 04536789 | James@yahoo.com | 37, leader street, Goodwood.  SA-5034 |  |  |  |  |  |  |
| P1==p2 |  |  |  |  |  |  |  |  | false | false |
| P1 == p3 |  |  |  |  |  |  |  |  | true | true |
| P1==p4 |  |  |  |  |  |  |  |  | true | false |
| P1.Address == p4.Address |  |  |  |  |  |  |  |  | true | true |

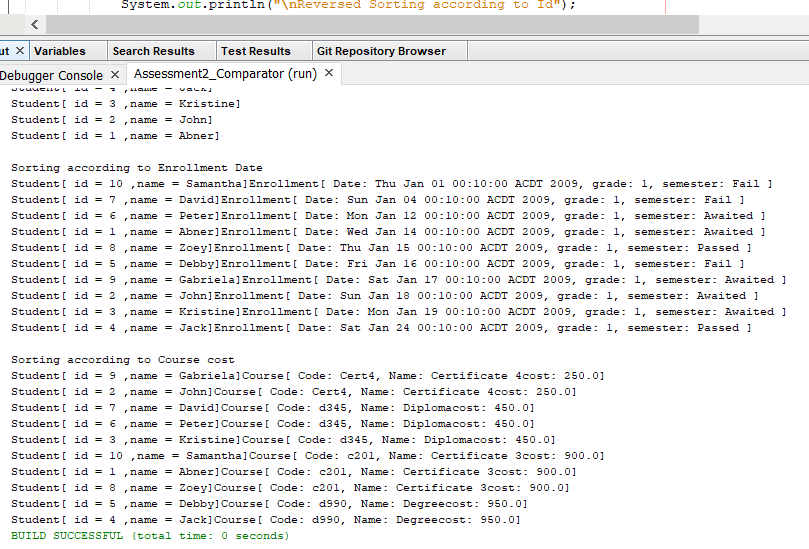
Junit Implemented for testcases of classes Person, Address, Student, Enrollment and, Course.

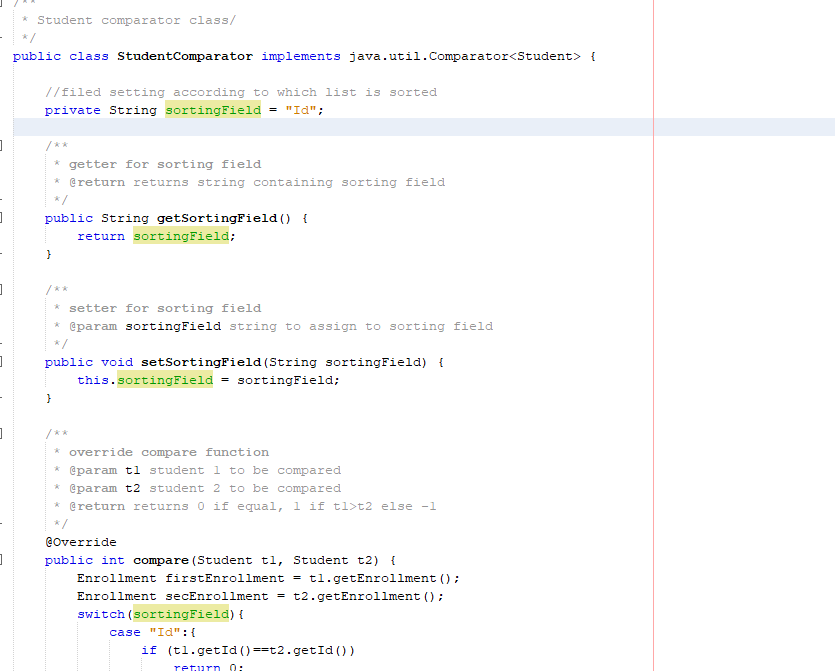
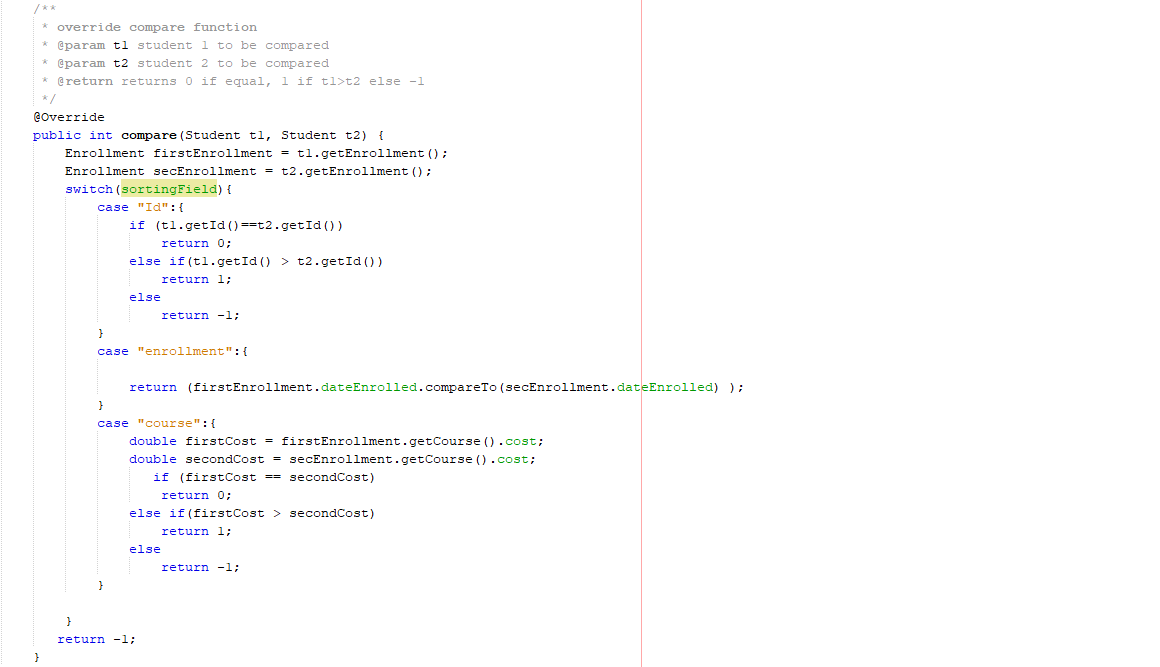
 

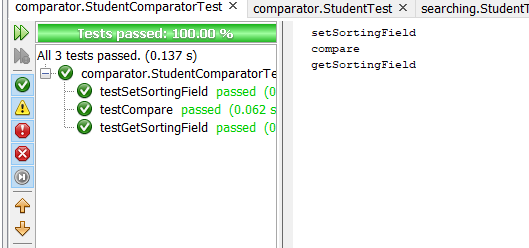
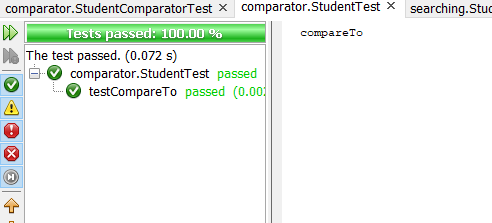
 

## Comparator





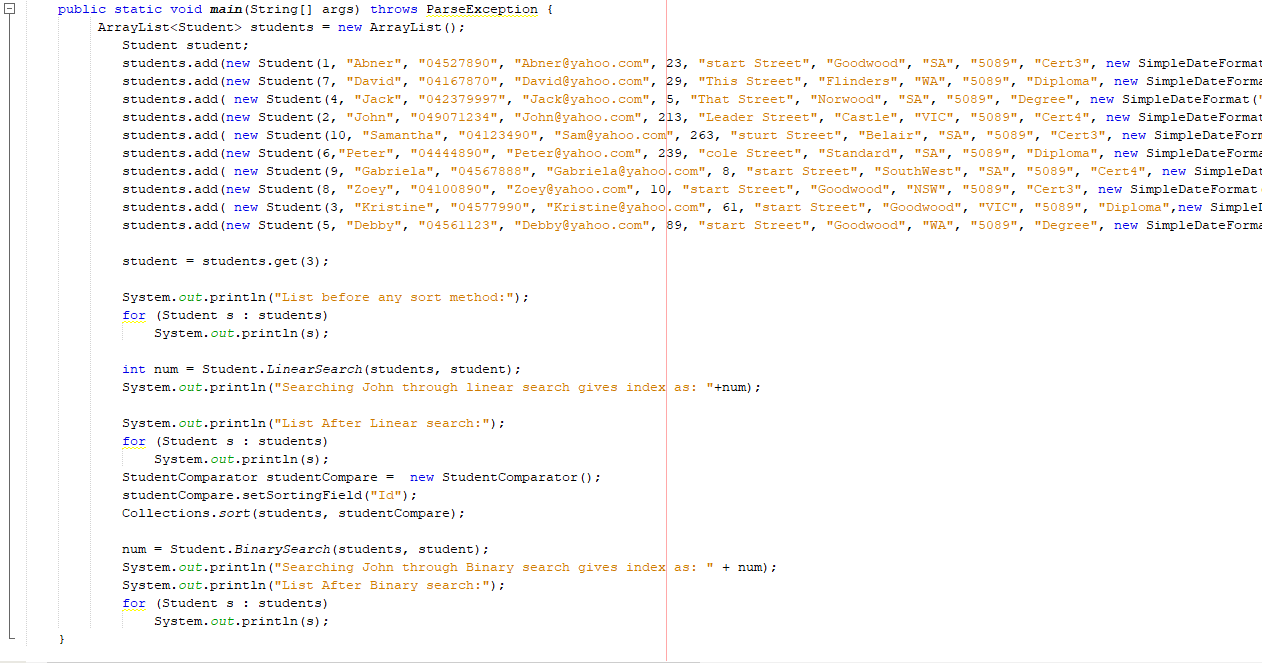
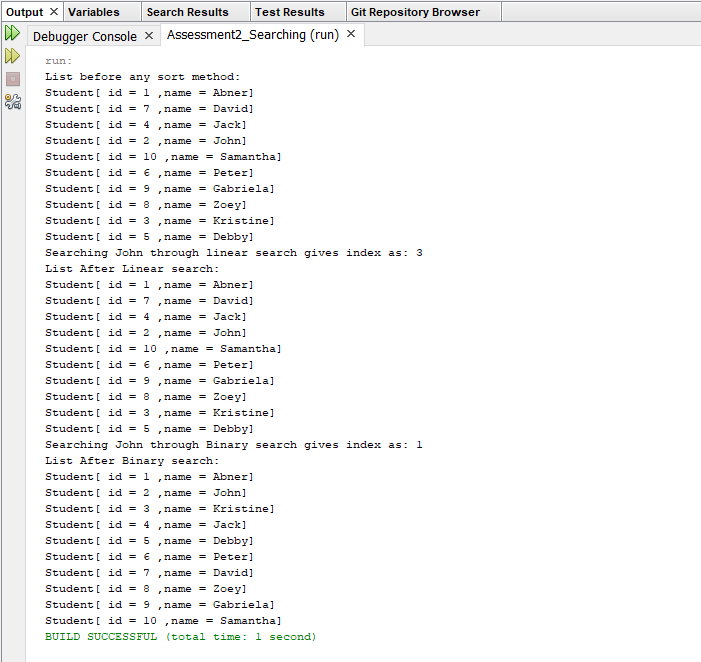
 

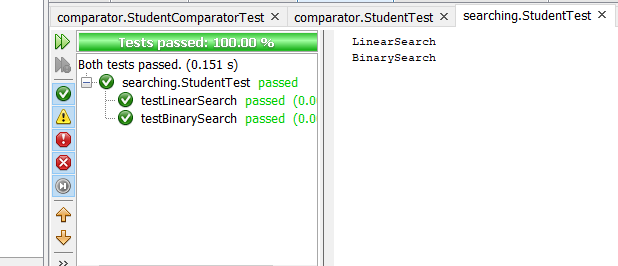
 

## Section C Searching

Searching Algorithms:

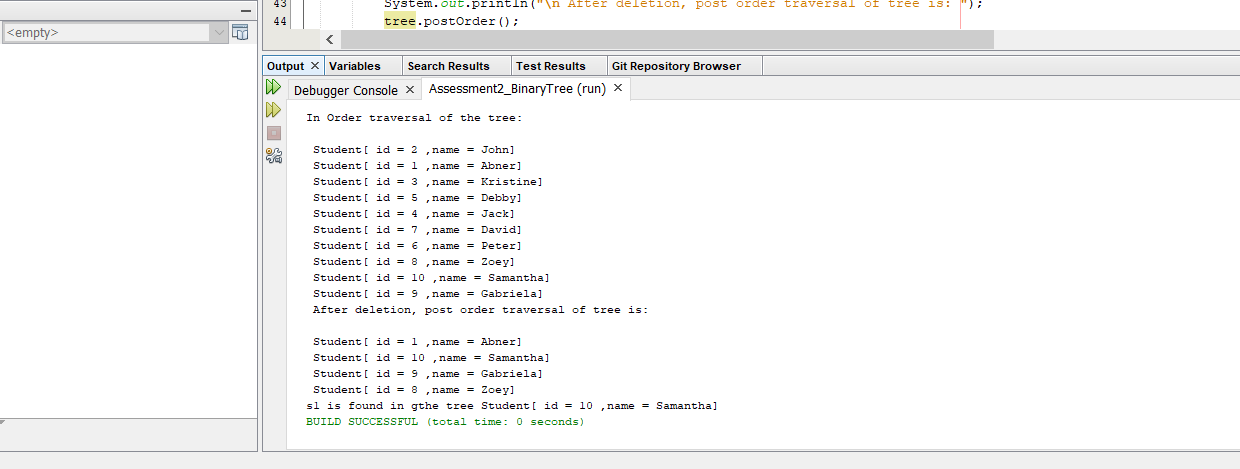
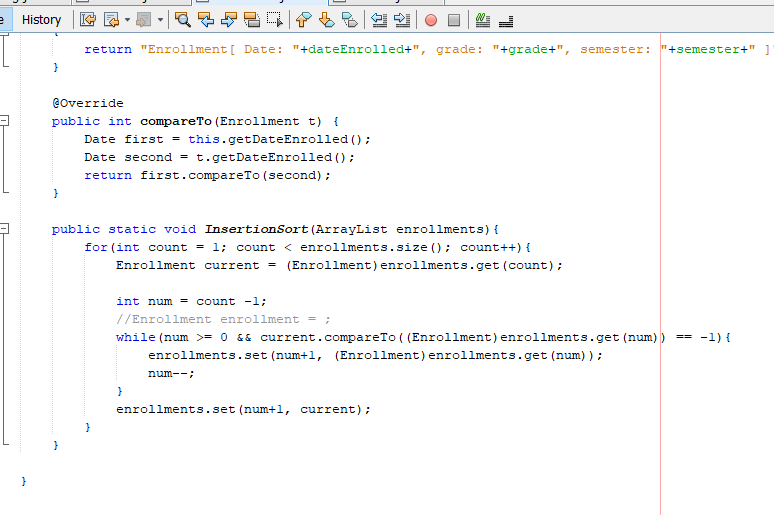
|  |  |  |
| --- | --- | --- |
| Algorithms | Pros | Cons |
| Linear search | Simple  Complexity is o(n)  Not affected by insertion or deletion | Traverses each element  Search time increases with increase in data |
| Binary Search | Search is done in half of the list | Complexity is o(log n)  Input needs to be sorted |
| Exponential Search | Works better than binary search for infinite array | Complexity is o(log n) |

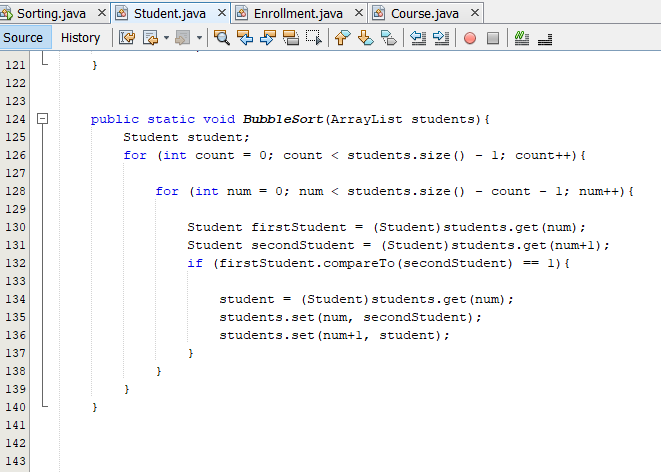
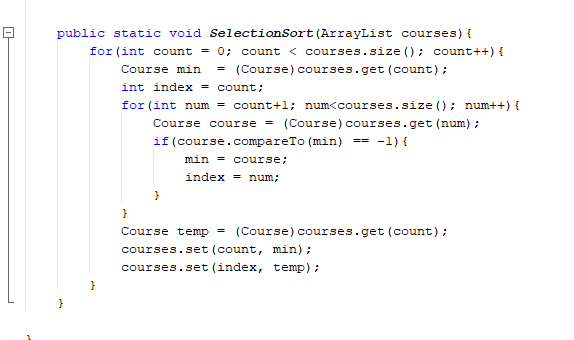
 

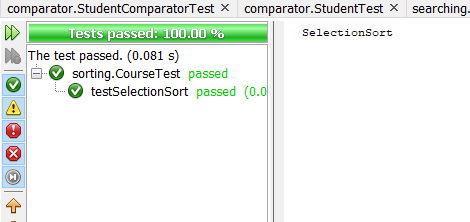
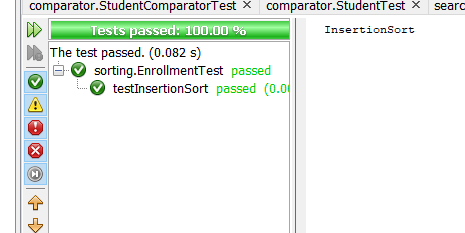


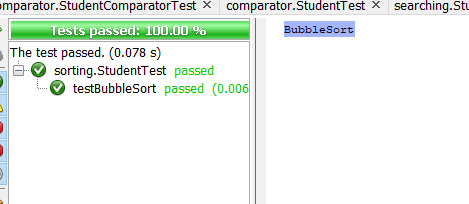
## Part D Sorting

Sorting Algorithms chosen: Selection sort implemented for course, Bubble sort, insertion sort

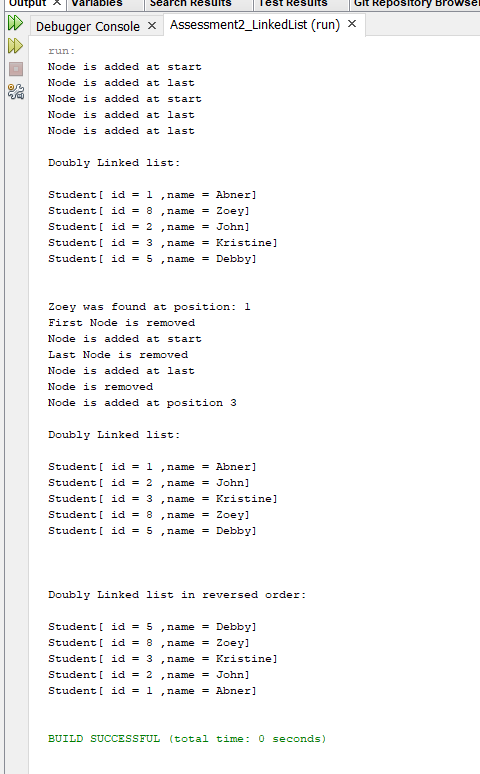
 



## E (Linked List)



## F Binary Tree

### 

## JavaDoc

