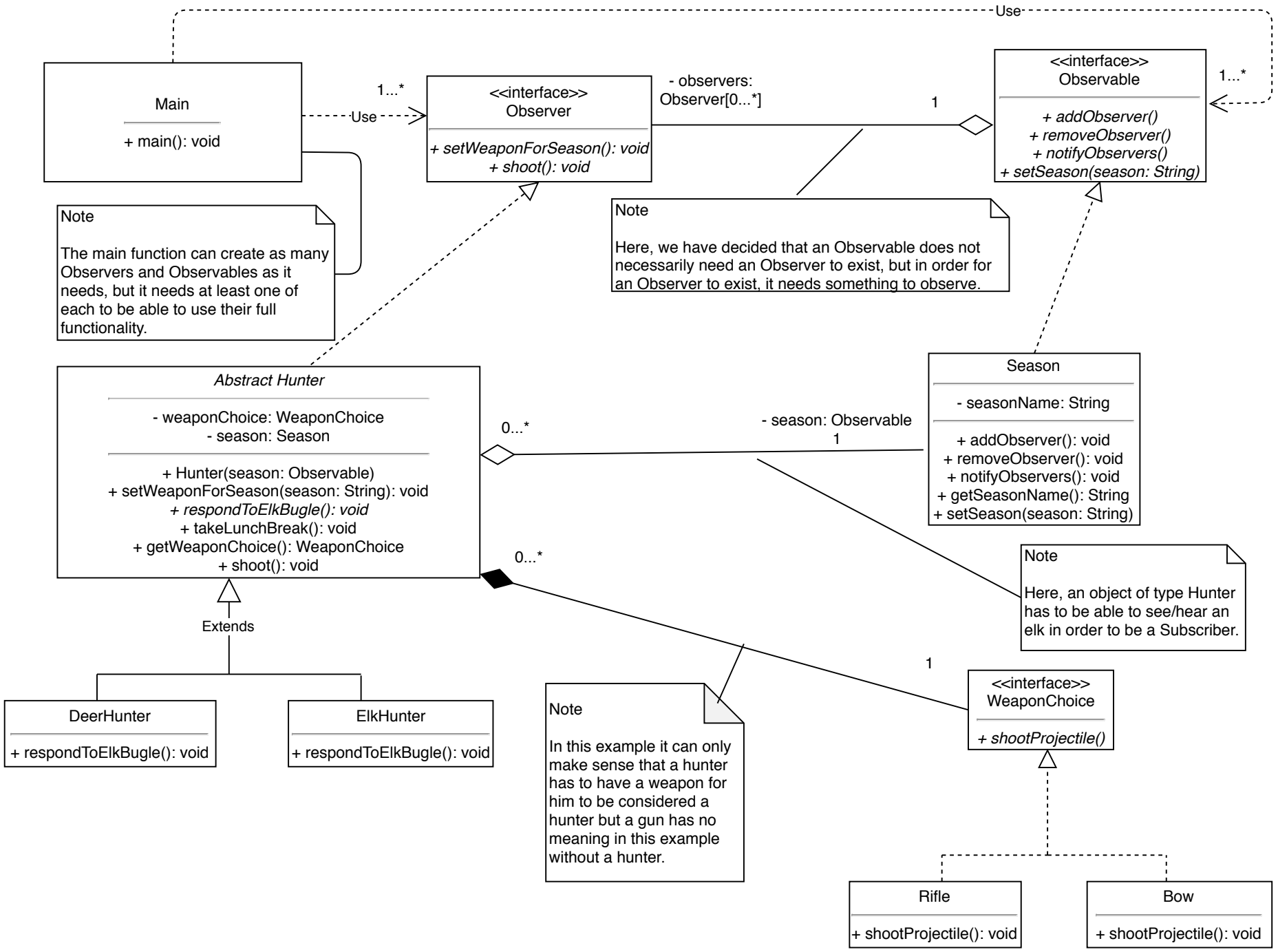
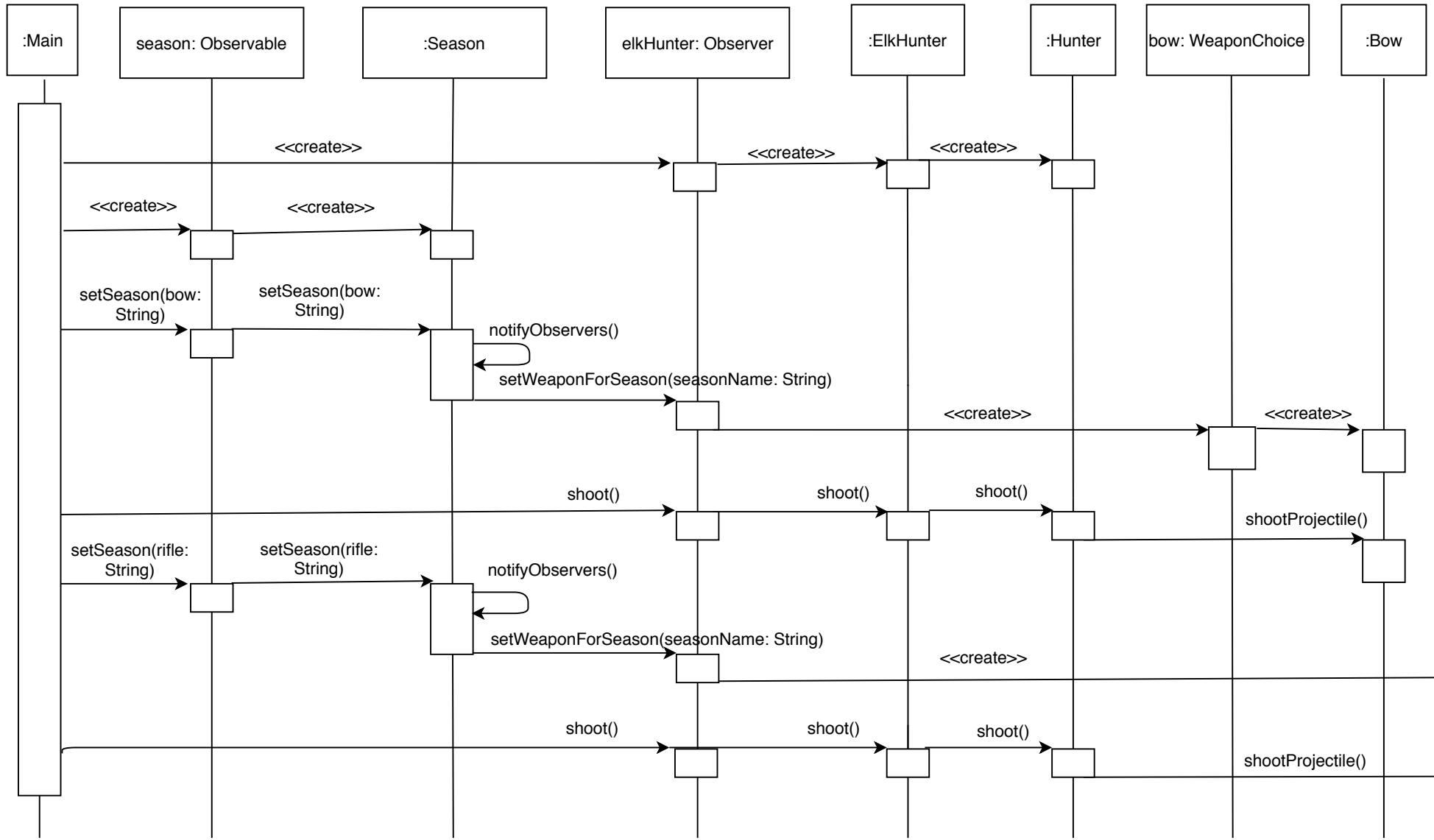
Daniel Gray
Tysen Radovich

Exercise
1A

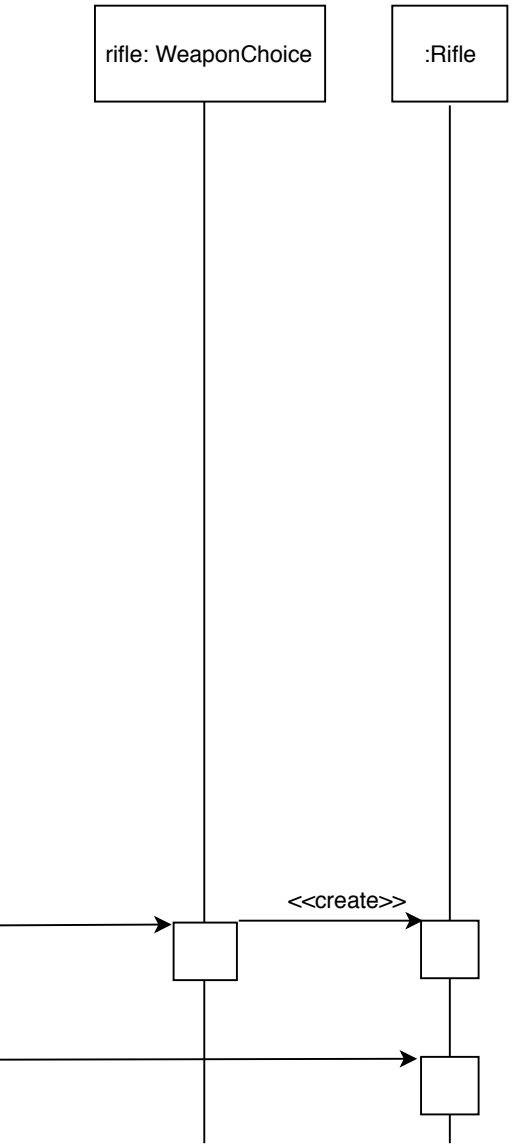


Exercise 1B
Part 1

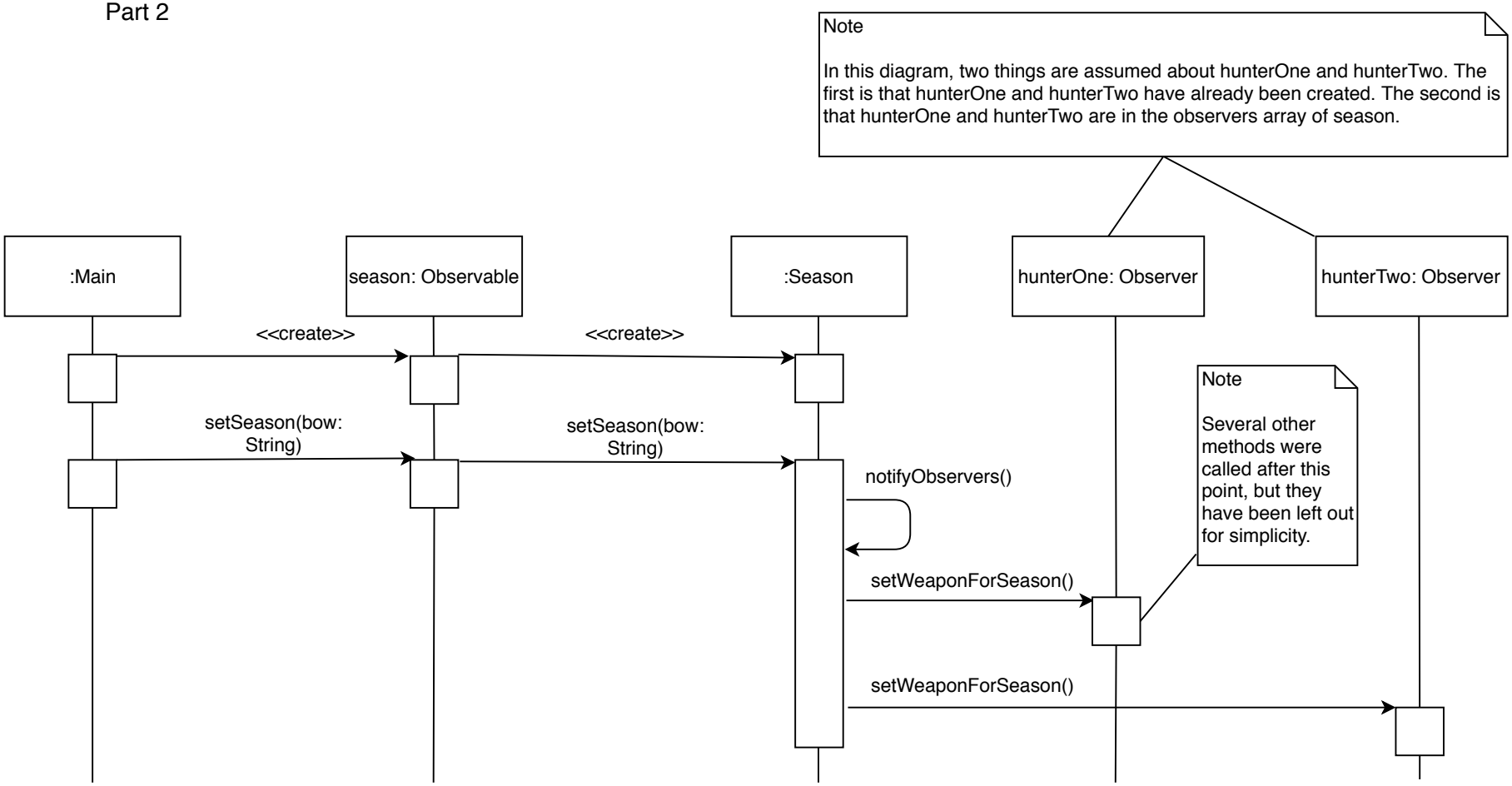


Note

This sequence diagram is an example of the Strategy pattern. This is because WeaponChoice is an interface, and the classes Bow and Rifle inherit from it. At runtime, we can change the hunter's weapon choice from Bow to Rifle or vice versa since we programmed to the interface. This allows us to change the behavior of shoot() and shootProjectile() at runtime based on the value of seasonName in the Season class..


rifle: WeaponChoice
:Rifle
<<create>>

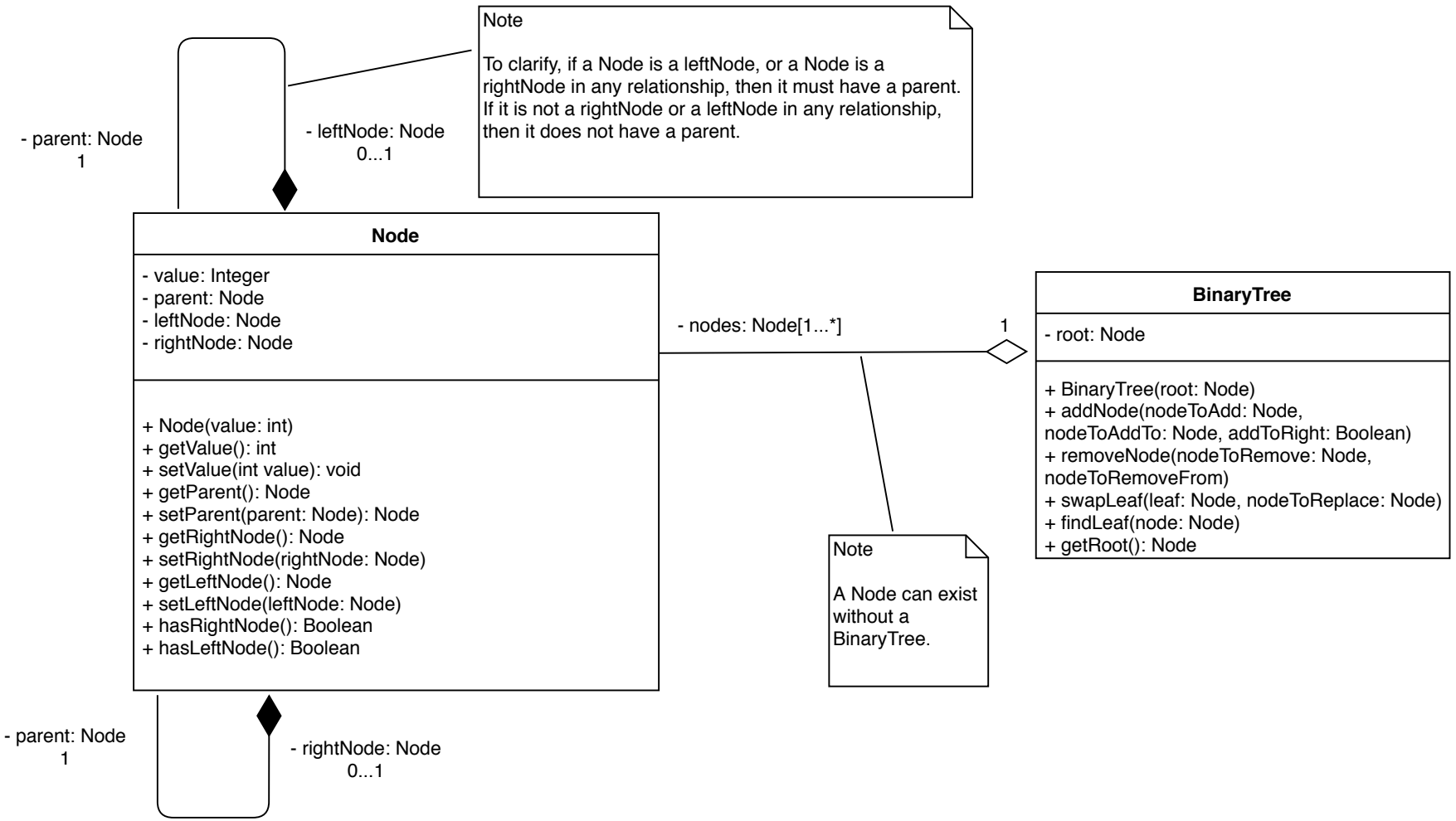
Exercise 1B
Part 2



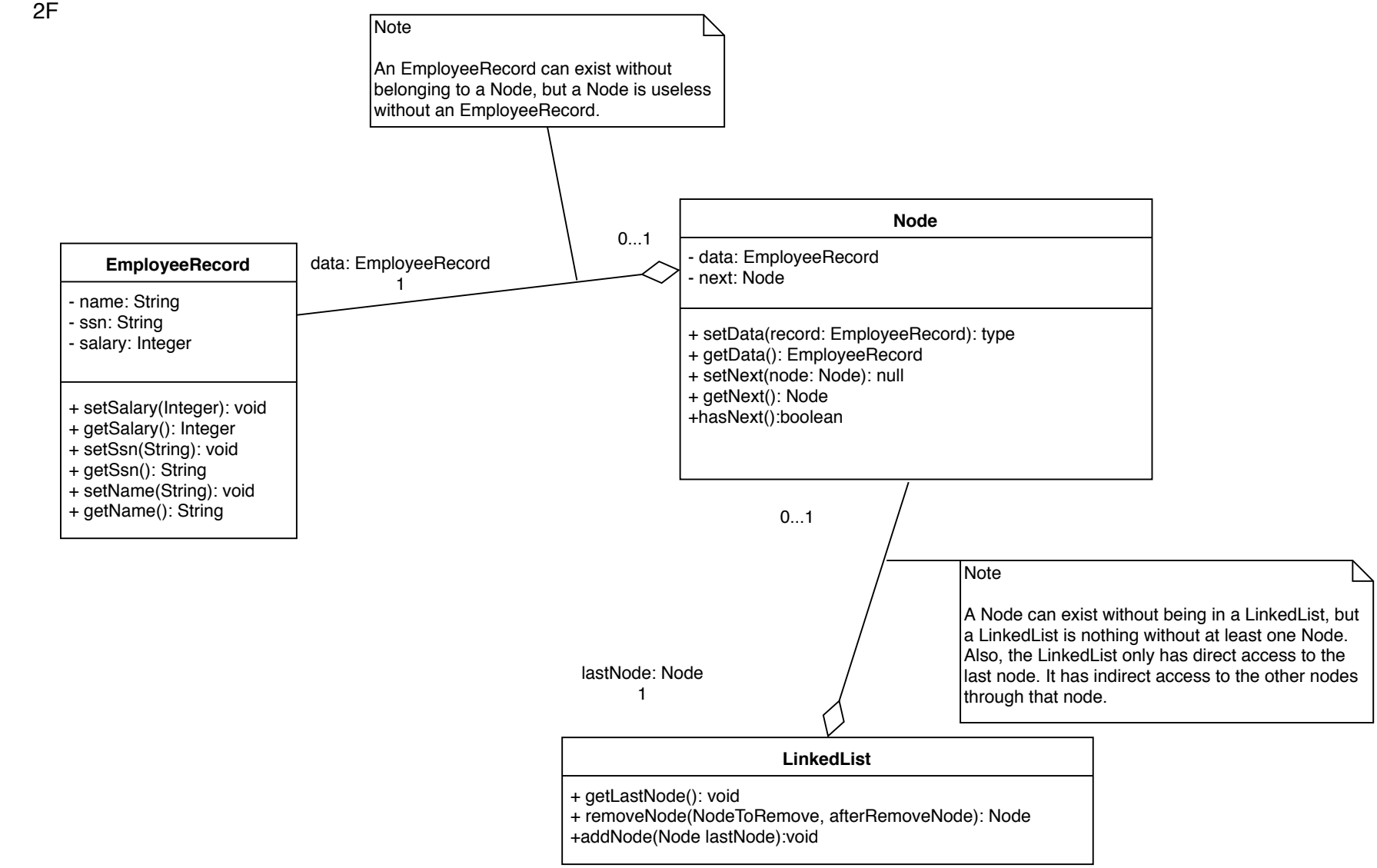
Note

This sequence demonstrates the Observer Pattern because hunterOne and hunterTwo are of type Observer, and are subscribed to season, which is of type Observable. We are able to send a notifyObservers() call to season, which then calls getSeasonName() on the Season class. This then updates hunterOne and hunterTwo via the setWeaponForSeason() calls.

Exercise
2D


Note
To clarify, if a Node is a leftNode, or a Node is a rightNode in any relationship, then it must have a parent. If it is not a rightNode or a leftNode in any relationship, then it does not have a parent.
- parent: Node
1
- leftNode: Node
0...1
Node
- value: Integer
- parent: Node
- leftNode: Node
- rightNode: Node
+ Node(value: int)
+ getValue(): int
+ setValue(int value): void
+ getParent(): Node
+ setParent(parent: Node): Node
+ getRightNode(): Node
+ setRightNode(rightNode: Node)
+ getLeftNode(): Node
+ setLeftNode(leftNode: Node)
+ hasRightNode(): Boolean
+ hasLeftNode(): Boolean
- nodes: Node[1...*]
1
BinaryTree
- root: Node
+ BinaryTree(root: Node)
+ addNode(nodeToAdd: Node, nodeToAddTo: Node, addToRight: Boolean)
+ removeNode(nodeToRemove: Node, nodeToRemoveFrom)
+ swapLeaf(leaf: Node, nodeToReplace: Node)
+ findLeaf(node: Node)
+ getRoot(): Node
Note
A Node can exist without a BinaryTree.
- parent: Node
1
- rightNode: Node
0...1

Exercise
2F


Note
An EmployeeRecord can exist without belonging to a Node, but a Node is useless without an EmployeeRecord.
EmployeeRecord
- name: String
- ssn: String
- salary: Integer
+ setSalary(Integer): void
+ getSalary(): Integer
+ setSsn(String): void
+ getSsn(): String
+ setName(String): void
+ getName(): String
data: EmployeeRecord
1
0...1
Node
- data: EmployeeRecord
- next: Node
+ setData(record: EmployeeRecord): type
+ getData(): EmployeeRecord
+ setNext(node: Node): null
+ getNext(): Node
+hasNext():boolean
0...1
Note
A Node can exist without being in a LinkedList, but a LinkedList is nothing without at least one Node. Also, the LinkedList only has direct access to the last node. It has indirect access to the other nodes through that node.
lastNode: Node
1
LinkedList
+ getLastNode(): void
+ removeNode(NodeToRemove, afterRemoveNode): Node
+addNode(Node lastNode):void