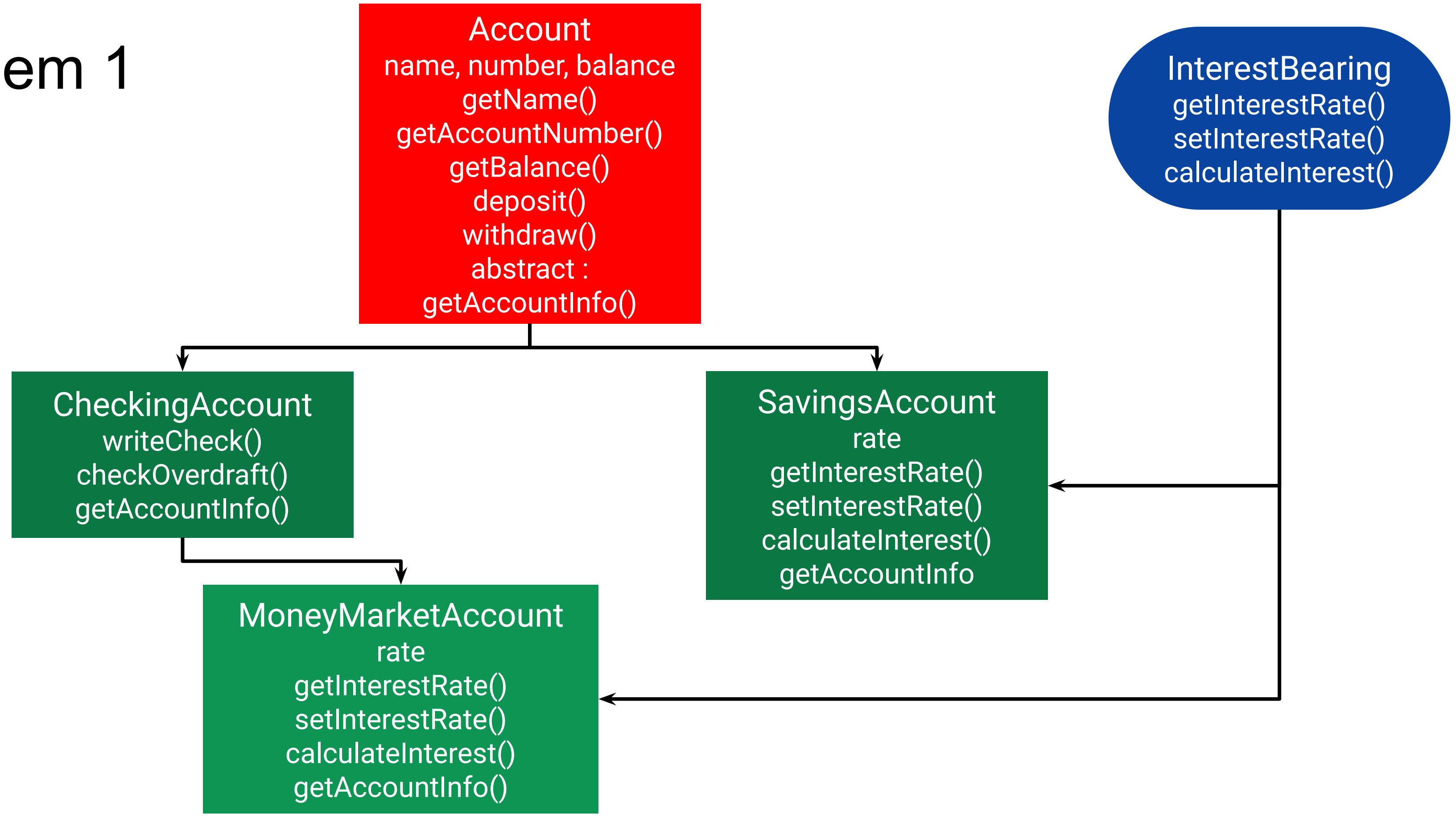
Exam I: Review

Fall 2018 Exam I

Announcements

- Tuesday (11:59 pm) - Thursday (11:59 pm)
- 2 parts
  - Zybooks (40 pts)
  - Canvas (60 pts)
- Topics: Inheritance, Polymorphism, Encapsulation, Static/Dynamic type, Compile-time errors, Run-time Exceptions, Primitive/Reference data type, equals(), clone(), shallow/deep copying, BigO, sorting - Insertion/Selection Sort.
- ~~Merge/Quick Sort, comparator, comparable~~

Problem 1


Account
name, number, balance
getName()
getAccountNumber()
getBalance()
deposit()
withdraw()
abstract :
getAccountInfo()
InterestBearing
getInterestRate()
setInterestRate()
calculateInterest()
CheckingAccount
writeCheck()
checkOverdraft()
getAccountInfo()
SavingsAccount
rate
getInterestRate()
setInterestRate()
calculateInterest()
getAccountInfo
MoneyMarketAccount
rate
getInterestRate()
setInterestRate()
calculateInterest()
getAccountInfo()

Problem 3c

```
static boolean hasI(int[] arr, int i) {
        if(i == arr.length)
                return false;
        if(arr[i] == 'i')     // ‘i’ = 105
                return true;
        else
                return hasI(arr, i+1);
  }
```

Problem 3c

arr =

100	102	104	106

arr.length = 4

```
static boolean hasI(int[] arr, int i) {
        if(i == arr.length)
                return false;
        if(arr[i] == 'i')     // ‘i’ = 105
                return true;
        else
                return hasI(arr, i+1);
   }
```

Problem 3c

```
static boolean hasI(int[] arr, int i) {
        if(i == arr.length)
                return false;
        if(arr[i] == 'i')     // ‘i’ = 105
                return true;
        else
                return hasI(arr, i+1);
  }
```

arr =

100	102	104	106

arr.length = 4

hasI(arr, 0)

Problem 3c

```
static boolean hasI(int[] arr, int i) {
        if(i == arr.length)
                return false;
        if(arr[i] == 'i')      // ‘i’ = 105
                return true;
        else
                return hasI(arr, i+1);
    }
```

arr =

🟥 100	102	104	106

arr.length = 4

hasI(arr, 0)

```
i = 0 (!= 4), arr[0] = 100 (!= 105)
return hasI(arr, 1)
```

Problem 3c

```
static boolean hasI(int[] arr, int i) {
        if(i == arr.length)
                return false;
        if(arr[i] == 'i')     // ‘i’ = 105
                return true;
        else
                return hasI(arr, i+1);
   }
```

arr =

100	102	104	106

arr.length = 4

hasI(arr, 0)

```
i = 0 (!= 4), arr[0] = 100 (!= 105)
return hasI(arr, 1)
        i = 1 (!= 4), arr[1] = 102 (!= 105)
        return hasI(arr, 2)
```

Problem 3c

```
static boolean hasI(int[] arr, int i) {
        if(i == arr.length)
                return false;
        if(arr[i] == 'i')     // ‘i’ = 105
                return true;
        else
                return hasI(arr, i+1);
   }
```

arr =

100	102	104	106

arr.length = 4

hasI(arr, 0)

```
i = 0 (!= 4), arr[0] = 100 (!= 105)
return hasI(arr, 1)
    i = 1 (!= 4), arr[1] = 102 (!= 105)
    return hasI(arr, 2)
        i = 2 (!= 4), arr[2] = 104 (!= 105)
        return hasI(arr, 3)
```

Problem 3c

```
static boolean hasI(int[] arr, int i) {
        if(i == arr.length)
                return false;
        if(arr[i] == 'i')     // ‘i’ = 105
                return true;
        else
                return hasI(arr, i+1);
   }
```

arr =

100	102	104	106

arr.length = 4

```
hasI(arr, 0)
    i = 0 (!= 4), arr[0] = 100 (!= 105)
    return hasI(arr, 1)
        i = 1 (!= 4), arr[1] = 102 (!= 105)
        return hasI(arr, 2)
            i = 2 (!= 4), arr[2] = 104 (!= 105)
            return hasI(arr, 3)
                i = 3 (!= 4), arr[3] = 106 (!= 105)
                return hasI(arr, 4)
```

Problem 3c

```
static boolean hasI(int[] arr, int i) {
        if(i == arr.length)
                return false;
        if(arr[i] == 'i')     // ‘i’ = 105
                return true;
        else
                return hasI(arr, i+1);
    }
```

arr =

100	102	104	106

arr.length = 4

hasI(arr, 0)

> i = 0 (!= 4), arr[0] = 100 (!= 105)
>
> return hasI(arr, 1)
>
> > i = 1 (!= 4), arr[1] = 102 (!= 105)
> >
> > return hasI(arr, 2)
> >
> > > i = 2 (!= 4), arr[2] = 104 (!= 105)
> > >
> > > return hasI(arr, 3)
> > >
> > > > i = 3 (!= 4), arr[3] = 106 (!= 105)
> > > >
> > > > return hasI(arr, 4)
> > > >
> > > > > i = 4 (== arr.length)
> > > > >
> > > > > return false

Insertion Sort

```
Iterate over all elements (i = 1 to i = n-1)
    While element to the left is bigger
        Swap
    // at the end of an iteration elements upto i are **locally** sorted
```

Selection Sort

```
Iterate over all elements (i = 0 to i = n-1)
    Starting at index i, find the index of the smallest element to the right
    Swap with element at i
    // at the end of an iteration elements upto i are globally sorted
```

Problem 4a: Insertion Sort

0	6	5	7	4	1	2	3
0	6	5	7	4	1	2	3
0	5	6	7	4	1	2	3
0	5	6	7	4	1	2	3
0	4	5	6	7	1	2	3
0	1	2	5	6	7	2	3
0	1	2	4	5	6	7	3
0	1	2	3	4	5	6	7

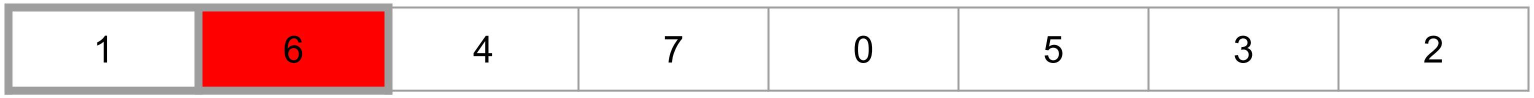
Problem 4d: Selection Sort

0	6	5	7	4	1	2	3
0	6	5	7	4	1	2	3
0	1	5	7	4	6	2	3
0	1	2	7	4	6	5	3
0	1	2	3	4	6	5	7
0	1	2	3	4	6	5	7
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2

Problem 4e: Insertion Sort



Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
🟩 1	🟩 6	4	7	0	5	3	2

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2
1	4	6	7	0	5	3	2

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2
1	4	6	🟥 7	0	5	3	2

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2
1	4	6	7	0	5	3	2
1	4	6	7	0	5	3	2

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2
1	4	6	7	0	5	3	2
1	4	6	7	0	5	3	2

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2
1	4	6	7	0	5	3	2
1	4	6	7	0	5	3	2
0	1	4	6	7	5	3	2

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2
1	4	6	7	0	5	3	2
1	4	6	7	0	5	3	2
0	1	4	6	7	5	3	2

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2
1	4	6	7	0	5	3	2
1	4	6	7	0	5	3	2
0	1	4	6	7	5	3	2
0	1	4	5	6	7	3	2

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2
1	4	6	7	0	5	3	2
1	4	6	7	0	5	3	2
0	1	4	6	7	5	3	2
0	1	4	5	6	7	3	2

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2
1	4	6	7	0	5	3	2
1	4	6	7	0	5	3	2
0	1	4	6	7	5	3	2
0	1	4	5	6	7	3	2
0	1	3	4	5	6	7	2

Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2
1	4	6	7	0	5	3	2
1	4	6	7	0	5	3	2
0	1	4	6	7	5	3	2
0	1	4	5	6	7	3	2
0	1	3	4	5	6	7	2

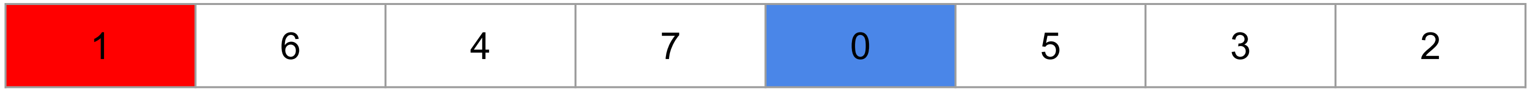
Problem 4e: Insertion Sort

1	6	4	7	0	5	3	2
1	6	4	7	0	5	3	2
1	4	6	7	0	5	3	2
1	4	6	7	0	5	3	2
0	1	4	6	7	5	3	2
0	1	4	5	6	7	3	2
0	1	3	4	5	6	7	2
0	1	2	3	4	5	6	7

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2

Problem 4e: Selection Sort



Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2
0	1	2	7	6	5	3	4

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2
0	1	2	7	6	5	3	4

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2
0	1	2	7	6	5	3	4
0	1	2	3	6	5	7	4

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2
0	1	2	7	6	5	3	4
0	1	2	3	6	5	7	4

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2
0	1	2	7	6	5	3	4
0	1	2	3	6	5	7	4
0	1	2	3	4	5	7	6

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2
0	1	2	7	6	5	3	4
0	1	2	3	6	5	7	4
0	1	2	3	4	5	7	6

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2
0	1	2	7	6	5	3	4
0	1	2	3	6	5	7	4
0	1	2	3	4	5	7	6
0	1	2	3	4	5	7	6

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2
0	1	2	7	6	5	3	4
0	1	2	3	6	5	7	4
0	1	2	3	4	5	7	6
0	1	2	3	4	5	7	6

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2
0	1	2	7	6	5	3	4
0	1	2	3	6	5	7	4
0	1	2	3	4	5	7	6
0	1	2	3	4	5	7	6
0	1	2	3	4	5	6	7

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2
0	1	2	7	6	5	3	4
0	1	2	3	6	5	7	4
0	1	2	3	4	5	7	6
0	1	2	3	4	5	7	6
0	1	2	3	4	5	6	7

Problem 4e: Selection Sort

1	6	4	7	0	5	3	2
0	6	4	7	1	5	3	2
0	1	4	7	6	5	3	2
0	1	2	7	6	5	3	4
0	1	2	3	6	5	7	4
0	1	2	3	4	5	7	6
0	1	2	3	4	5	7	6
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7