Week 13: Lecture 1, 2, 3, 8 Review

- Arrays, Objects, Classes
  - Array operations, functions, method overloading, strings, pass by value, immutability
- OOP Thinking, Inheritance, Polymorphism
  - OOP concepts, equals, method overriding
- Abstract Classes, Interfaces
  - Defining and implementing interfaces, extending abstract classes, comparable, comparator
- Developing Efficient Algorithms
  - Time/space complexity, basic sorting algorithms, logarithmic-time algorithms

Algorithm Complexity

- Short answer:

Consider this algorithm `algo`.

```
1.  public boolean algo(int[] a) {
2.      for (int i = 0; i < a.length; i++) {
3.          for (int j = i + 1; j < a.length; j++) {
4.              if (a[i] == a[j]) {
5.                  return true;
6.              }
7.          }
8.      }
9.      return false;
10. }
```

What is the time complexity of this algorithm?

Answer:

Algorithm Complexity

- Complete the blank:

The worst-case time complexity of the following algorithm is O( ).

```
1.  public int algo(int[] a, int k) {
2.      int n = a.length;
3.      for (int i = 0; i < n; i++) {
4.          if (a[i] == k) {
5.              return i;
6.          }
7.      }
8.      return -1;
9.  }
```

Lab Group 1 Q1, Q2, Q3, Q4, Q5, Q6

- Complete the class so that the Book class is **immutable**:

```
1  public class Book {
2      private String title;
3      Q1 String[] authors;
4
5      public Book(String title, Q2 authors) {
6          Q3 .title = title;
7          this.authors = new String[ Q4 ];
8          System.arraycopy(authors, 0, this.authors, 0, authors.length);
9      }
10
11     public Book(Book other) {
12         Q5 (other.title, other.authors);
13     }
14 }
```

- In line 11-13, we have an implementation of a/an Q6

Lab Group 2 Q1, Q2, Q3, Q4, Q5, Q6

- Complete the class so that the Person class is **immutable**:

```
1  public class Person {
2      private String name;
3      Q1 Date dateOfBirth;
4
5      public Person(String name, Date dateOfBirth) {
6          Q2 .name = name;
7          this.dateOfBirth = Q3 (dateOfBirth.getTime());
8      }
9
10     Q4 Date getDateOfBirth() {
11         return Q5 (dateOfBirth.getTime());
12     }
13 }
```

- In line 11, we use a technique called Q6 to return a fresh instance of an object instead of a reference to the instance variable

Lab Group 1 Q7, Q8, Q9

- Complete the classes:

```
1  class Employee {
2      void Q7 {
3          System.out.println("Employee is working");
4      }
5  }
6  class Manager Q8 Employee {
7      @Override
8      void work() {
9          Q9.work();
10         System.out.println("Manager is overseeing");
11     }
12 }
13 public class Program {
14     public static void main(String[] args) {
15         Manager manager = new Manager();
16         manager.work();
17     }
18 }
```

- Output:

```
Employee is working
Manager is overseeing
```

Lab Group 2 Q7, Q8, Q9

- Complete the classes:

```
1  class Person {
2      Q7 introduce() {
3          System.out.println("I am a person.");
4      }
5  }
6  class Student Q8 Person {
7      @Override
8      void introduce() {
9          Q9 ;
10         System.out.println("I am a student.");
11     }
12 }
13 public class Program {
14     public static void main(String[] args) {
15         Student student = new Student();
16         student.introduce();
17     }
18 }
```

- Output:

```
I am a person.
I am a student.
```

Lab Group 1 Q10, Q11, Q12

- Complete the class:

```
1  public class Book {
2      private String title;
3      private String author;
4
5      public Book(String title, String author) {
6          this.title = title;
7          this.author = author;
8      }
9
10     @Override
11     public boolean equals(Q10) {
12         if (obj == null) return false;
13         if (this == obj) return true;
14         if (!(obj instanceof Book)) return Q11
15         Book book = (Book) obj;
16         return Q12
17     }
18 }
```

Lab Group 2 Q10, Q11, Q12

- Complete the class:

```
1  public class Employee {
2      private int id;
3      private String dept;
4
5      public Employee(int id, String dept) {
6          this.id = id;
7          this.dept = dept;
8      }
9
10     @Override
11     public boolean equals(Object that) {
12         if (that == null) return Q10
13         if (this == that) return true;
14         if (!( Q11 )) return false;
15         Employee thatEmp = (Employee) that;
16         return Q12
17     }
18 }
```

Lab Group 1 Q13, Q14, Q15

- Complete the class:

```
1  public class Employee implement Q13 {
2      private double salary;
3
4      @Override
5      public int compareTo(Object obj) {
6          Employee that = (Employee) obj;
7          if (this.salary > that.salary)
8              return 1;
9          if ( Q14 )
10             return 0;
11         return Q15
12     }
13 }
```

Lab Group 2 Q13, Q14, Q15

- Complete the class:

```
1  public class Person implements Comparable {
2      private int age;
3
4      @Override
5      public int compareTo( Q13 ) {
6          Person person = (Person) other;
7          if (this.age > person.age)
8              return 1;
9          if ( Q14 )
10             return -1;
11         return Q15
12     }
13 }
```