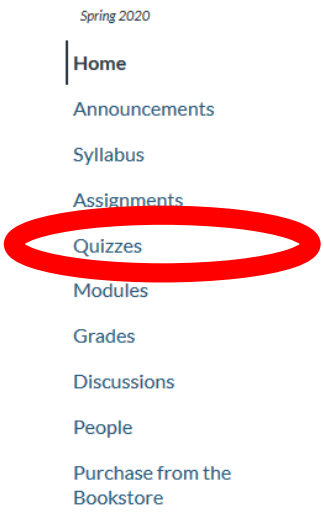

EECS 293:

Quiz Starting at 10:35am
Head to canvas and click on Quizzes
You can start as soon as the quiz is published on canvas
Lecture starting soon ...

1 minute left ...

Class Design: Containment and Inheritance

Containment

```
class TypeName {
    private String identifier;
...
}
```

TypeName contains an identifier

Two main types:

- Composition (e.g., university is composed of departments)
- Aggregation (e.g., department is an aggregation of profs, students, staff)

Questionable: more than 9 data members

Inheritance

☺ class AbstractTypeEntry implements TypeEntry
? class SimpleTypeEntry extends AbstractTypeEntry

Main concept: models **is-a** relationship
E.g. AbstractTypeEntry is a TypeEntry

```
class EmployeeCensus extends ArrayList<Employee> {
    ...
}
```

Benefit: some methods for free, e.g.,

```
for (var e: employeeCensus) { ... }
```

Problems:

- Extends (Item 18)
- Nothing in EmployeeCensus that is an array-backed list: **inconsistent abstraction**

```
class EmployeeCensus implements Iterable<Employee> {
    Set<Employee> roster = ...;

    // to implement for-each
    @Override
    Iterator<Employee> iterator() {
        return roster.iterator();
    }
```

Liskov Substitution Principle

```
class BasicType implements Type { ... }

Type t = new BasicType(); // OR ...
Type t = new CompoundType(); // OR ...
```