

Exam Reminders

Prelim 1 is Tonight!

Early Exam: 5:30-7:00, Statler 196

Main Exam: 7:30-9:00, Statler 185

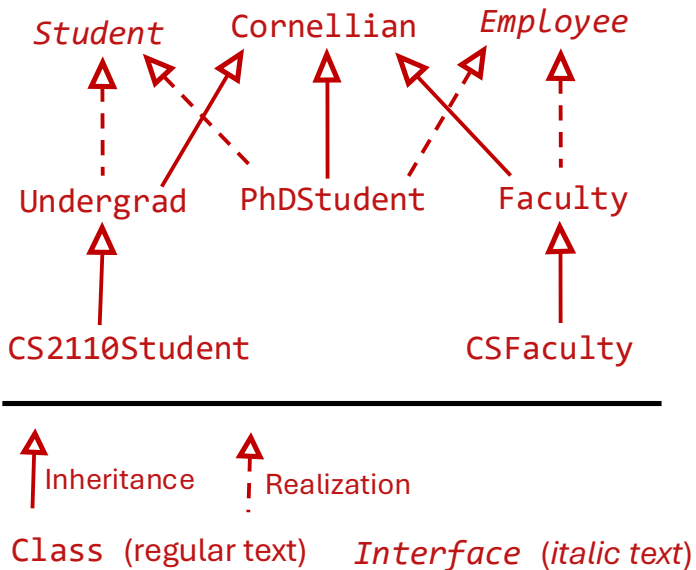
Bring your **Cornell ID Card** and a couple **writing utensils** (pencils, erasers, pens)
Exam is closed-book

More information and review materials linked on website / Ed

You've learned a lot so far! Time to show it off!

Type Hierarchies and Single Inheritance

Interfaces and inheritance let us build deep, complex type hierarchies



Java's single-inheritance rule:

Every* class has exactly one parent class (i.e., immediate superclass).



Lecture 11: Additional Java Features

CS 2110

September 29, 2026

Today's Learning Outcomes

- 21. Explain exceptions and their relationship to specifications and defensive programming.
- 22. Write code that throws, propagates, and handles exceptions.
- 46. Explain the concept of *immutability* and its relation to the final keyword in Java.
- 47. Contrast the == operator and the equals() method, and determine the appropriate one for a given scenario.

Part 1: Exception Handling

Exceptional Behavior

Sometimes, a program may encounter a situation that prevents it from proceeding with its normal control flow.

What should we do?

Exception Handling in Java

An **exception** is an object that signals that a program has encountered an atypical circumstance.

Java has special language features and syntax for working with exceptions.

The throw Statement

When you detect a situation that your code is unable to handle, use a throw statement to generate an exception.

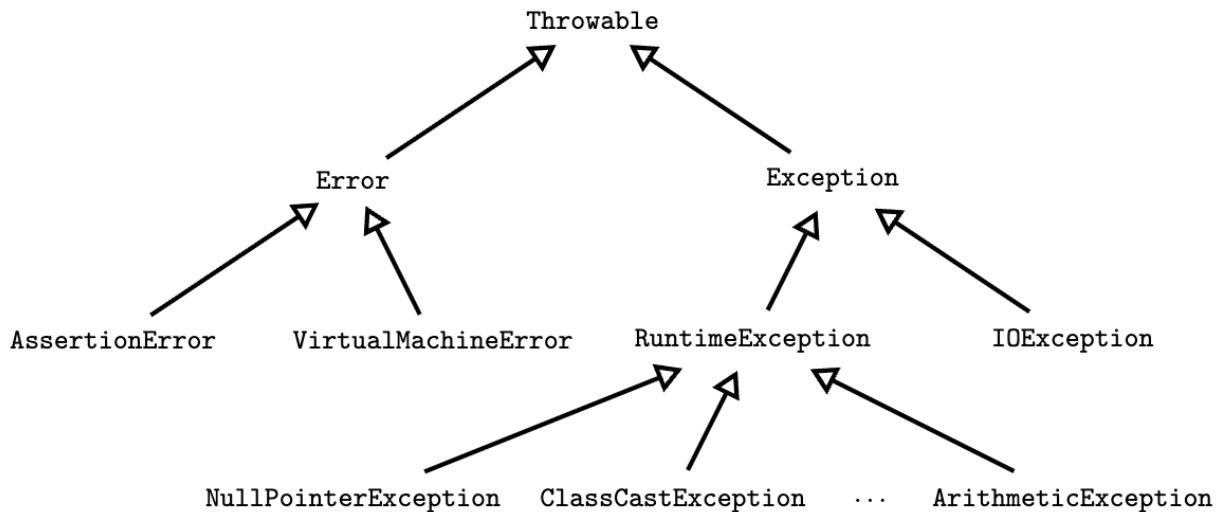
throw statements stop normal control flow and **propagate** down the runtime stack until they are **caught**



Coding Demo: Custom Exceptions



The Throwable Hierarchy



Checked Exceptions and throws Clauses

If a method has the possibility of *propagating* (i.e., either throwing itself or passing along from its own call) a *checked* exception, it must disclose this in a `throws` clause attached to the method signature.

try/catch Blocks

New syntax:

Exceptions and Specifications

Subtle interplay between these (frequent source of misconceptions)

1. Undefined Behavior may or may not result in an exception.

2. Exceptions are not undefined behavior.

Since exceptions are part of specs, we write unit tests that check for them:

JUnit's `assertThrows()` method

Part 2: Immutability and final

Immutability

A class is **mutable** if modifications can be made to its objects' state after their construction. Otherwise, the class is **immutable**.



Coding Demo: The Point Class



The `final` Keyword

Recall: `final` methods cannot be overridden in subclasses.

Part 3: The Object Class

The super-est Class

The Object class is the top of Java's type hierarchy.

- Object is the default superclass of any class whose declaration does not include an extends clause.
- Thus, every Java object is an Object.

The Object class contains a few basic methods that are defined for every reference type.

Today: `toString()` : allows every object to be printed
 `equals()` : allows any pair of objects to be compared

 (more coming soon...)



Coding Demo: Overriding Object Methods



Java's Two Notions of Equality

`==` : Compares the **values** of the expressions on both sides

`equals()` : Allows objects to define a way to compare their **states**

A Good equals() Template

```
@Override
public boolean equals(Object obj) {
    if ((obj == null) || (this.getClass() != obj.getClass())) {
        return false;
    }

    // replace this to a cast of your object type
    Point other = (Point) obj;

    // compare all fields (== for primitive types, equals() for reference types)
    return (this.x == other.x) && (this.y == other.y);
}
```



Coding Demo: record Classes

