Exam 1 Review Topics
Exam technical details

• When: Monday, October 4th in class
• Bring your student ID
• Seats are assigned
• Up to two pages of notes allowed
  – 8.5x11 paper (double sided is fine). Typed or handwritten.
• No electronic devices
  – Including calculators, watches, iwatches, phones, laptops, tablets, ...
• Multiple choice
• Can grade as you exit the exam
Objects and Classes/Review of 1323

• What is an object?
• How do you create one?
• What is a constructor and how do you use one?
• What is a method and how do you use it?
• What is the state of an object?
• Trace through a program’s execution
Object Oriented Design

- Abstraction and Encapsulation
  - Abstract away the implementation details
  - Users of your classes shouldn’t need to know the implementation

- Aggregation
  - Has-a

- Immutable vs mutable classes
UML

Know how to interpret and compare class diagrams

• Hierarchies, abstract classes, aggregation
• Listing all variables and methods properly, including visibility and static/instance
Input/Output

• How to read a line from a BufferedReader?
• How to parse the resulting String?
• How to interpret substrings in terms of ints and doubles?
• Using System.out.println()

You should be able to recognize the syntax of opening a file vs System.in, but you won’t need to generate the syntax
Inheritance and Polymorphism

• What does it mean to inherit from a superclass?
  – What methods and variables can you access?
  – Is-a relationship

• Overloading versus Overriding
  – Overloading has the same name but different parameters
  – Overriding overwrites the superclass method (same name and parameters)

• super keyword

• Polymorphism allows you to use super and subtypes in reference declarations
**ArrayList**

- Array lists give you non-fixed arrays
- `ArrayList<E>`: `ArrayList` of a particular type of object
- Know how to use, including
  - Initialization
  - Adding items
  - Getting items
  - Iterating through the items
Exception and Error handling

• What is an exception?
• Why do you throw one?
• Exceptions versus Errors
• Exceptions that need to be caught versus not
  – RuntimeExceptions versus other Exceptions
• Try/catch/finally
Abstract Classes and Interfaces

• How does an abstract class differ from a regular superclass?
• Why use abstract classes?
• How do they differ from Interfaces?
  – Why use an interface?
• Comparable vs Comparator
Not on the Exam

• Java Generics and later topics

• I will not ask you to parrot back elements of the Java API
  – But you do need to be able to recognize and interpret the most common elements (that we have used)