All topics from first two exams and topic list, plus:

Write a complete program that demonstrates a simple use of Java's lambda. Make the lambda as compact as you can.

Write a complete program that demonstrates a use of Java's lambda where the method takes two parameters. Make the lambda as compact as you can.

Your Car class needs to be able to add, delete, and iterate over Passengers. Should you extend a List class or add a List as an attribute? As discussed in class, how do you go about making this decision? How would you code it?

What is the difference between a custom iterator and an explicit iterator? Code an example of each.

What achievement of Fred Brooks did we discuss in class?

How would you implement serialization if Java did not provide an interface and default implementation? What is the general purpose of serialization?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Design patterns:

MVC (esp model - view separation)

Strategy

Observer

Flyweight

Factory

Sets

hashCode

equals

Git add, commit, status, commit, log, branch, merge, pull

dispatch vector

switch (vs if)

Comparable

Comparator

sorting

equals, ==

toString

iterators

why we use setters/getters, and don't shadow vars

polymorphism

div/mod

arrays

Collection - HashSet, ArrayList

Collections

inline anonymous class

command line java

arrays

iterators

for each loop (enhanced for loop)

abstract classes

interfaces

Storyboards

Agile (Scrum flavor)

UML - class, sequence

Exceptions

javadoc

inheritance vs composition

serialization

lambda

enumerations