ArrayList class

• list of Objects with flexible sizing (like our FlexArray class examples but with more methods)

• How deal with problem of having to rewrite code depending on the type of item in the array?
  
  for example we had StackOfInts and StackOfChars, FlexArray of Reviews, but then we couldn’t use it for FlexArray of Strings to hold reviewerNames.

• want just one class that accept the different objects without rewriting, but where the compiler can still check that you are using it correctly. **Solution is to use generics**

• Documentation uses: `java.util.ArrayList<E>` where E is a **placeholder** for the type

• Invoke with: `ArrayList a<String>. = new ArrayList<String>` with desired type inserted (must be of type object, not primitive variable)

• can have many ArrayLists in same program with different types,
Exceptions

• finally we are getting to the “nice” way to handle runtime errors. Instead of program just crashing, or exiting perhaps with a printing something, we can control the way the program handles errors (*index out of bounds, null pointer, arithmetic, input mismatch …*)

• Terminology:
  • an exception is an object that represents an error
    e.g. you can say `new ArithmeticException("message goes here");`
  • an exception is *thrown*
    e.g. `throw new ArithmeticException("message goes here");`
  • it must be *handled* in some way: either *caught* or *rethrown* (thrown up the calling chain to another method)
  • to catch an exception, use *try-catch blocks*

see demos *Quotient1* through *Quotient6.java*
• Runtime exceptions (index bounds, null pointer, arithmetic) are **unchecked** (too unwieldy to require try/catch block for every array index)

• You can create your own exception too
Exceptions

• Every method must state kinds of checked exceptions it might throw: called declaring the exception
e.g. public void myMethod() throws IOException, Exception2;
(subclass can’t throw Exception unless Superclass does)

• When error detected:
  IllegalArgumentException ex = new IllegalArgumentException(“wrong arg”);  or equivalently
  throw new IllegalArgumentException(“message”);

• When exception thrown, it can be caught and handled. Also can catch more than one exception
  try {
    statements;
  }
  catch (Exception1 ex1){
    handle ex1;
  }
  catch (Exception2 ex2){
    handle ex2;
  }

  also
  try {
    statements;
  }
  catch (Exception1 ex1| Exception2 ex2){
    handle ex1;
  }

  order from most specific exception to least

• If statement in try block throws an exception, execution goes right to appropriate catch block and
does not finish the rest of try statements.

• The exception handler goes up the call stack from current method to calling method to find a handler

• Can get information from exception: getMessage(), printStackTrace(), more …