```
class Report
  def Report.report(out, machines, robot)
   out.print "FACTORY REPORT\n"
   machines.each do |machine|
     out.print "Machine #{machine.name}"
    out.print "bin=#{machine.bin}" if machine.bin != nil
    out.print "\n"
  end
  out.print "\n"
  out.print "Robot"
  if robot.location != nil
    out.print "location=#{robot.location.name}"
  end
  out.print "bin=#{robot.bin}" if robot.bin != nil
  out.print "\n"
  out.print "========\n"
  end
end
```

(In the code download you can find Rspec examples showing how these classes interact.)

A. In `Report.report`, circle four blocks of code to show which functions you might extract in the process of refactoring this code.

B. Rewrite the `report` method as four statements, as if you had done *Extract Method* for each block.

C. Does it make sense to extract a one-line method?

See page 217 for solution ideas.

Exercise 5.3: Large Class

Consider the API for the String class in Ruby 1.8.6:

```
str % arg
str * integer
str + integer
str << fixnum
str << obj
str.concat(fixnum)
str.concat(obj)
```

```
str <=> other_str
str == obj
str =~ obj
str[fixnum]
str[fixnum, fixnum]
str[range]
str[regexp]
str[regexp, fixnum]
str[other_str]
str[fixnum] = fixnum
str[fixnum] = new_str
str[fixnum, fixnum] = new_str
str[range] = aString
str[regexp] = new_str
str[regexp, fixnum] = new_str
str[other_str] = new_str
str.capitalize
str.capitalize!
str.casecmp(other_str)
str.center(integer, padstr)
str.chomp(separator=$/)
str.chomp!(separator=$/)
str.chop
str.chop!
str.concat(fixnum)
str.concat(obj)
str.count([other_str]+)
str.crypt(other_str)
str.delete([other_str]+)
str.delete!([other_str]+>)
str.downcase
str.downcase!
str.dump
str.each(separator=$/) {|substr| block }
str.each_byte {|fixnum| block }
str.each_line(separator=$/) {|substr| block }
str.empty?
str.eql?(other)
str.gsub(pattern, replacement)
str.gsub(pattern) {|match| block }
str.gsub!(pattern, replacement)
str.gsub!(pattern) {|match| block }
str.hash
str.hex
str.include? other_str
str.include? fixnum
```

```
str.index(substring [, offset])
str.index(fixnum [, offset])
str.index(regexp [, offset])
str.insert(index, other_str)
str.inspect
str.intern
str.length
str.ljust(integer, padstr=' ')
str.lstrip
str.lstrip!
str.match(pattern)
str.next
str.next!
str.oct
str.replace(other_str)
str.reverse
str.reverse!
str.rindex(substring [, fixnum])
str.rindex(fixnum [, fixnum])
str.rindex(regexp [, fixnum])
str.rjust(integer, padstr=' ')
str.rstrip
str.rstrip!
str.scan(pattern)
str.scan(pattern) {|match, ...| block }
str.slice(fixnum)
str.slice(fixnum, fixnum)
str.slice(range)
str.slice(regexp)
str.slice(regexp, fixnum)
str.slice(other_str)
str.slice(fixnum)
str.slice(fixnum, fixnum)
str.slice(range)
str.slice(regexp)
str.slice(regexp, fixnum)
str.slice(other_str)
str.slice!(fixnum)
str.slice!(fixnum, fixnum)
str.slice!(range)
str.slice!(regexp)
str.slice!(other_str)
str.split(pattern=$;, [limit])
str.squeeze([other_str]*)
str.squeeze!([other_str]*)
str.strip
```

```
str.strip!
str.sub(pattern, replacement)
str.sub(pattern) {|match| block }
str.sub!(pattern, replacement)
str.sub!(pattern) {|match| block }
str.succ
str.succ!
str.sum(n=16)
str.swapcase
str.swapcase!
str.to_f
str.to_i(base=10)
str.to_s
str.to_str
str.to_sym
str.tr(from_str, to_str)
str.tr!(from_str, to_str)
str.tr_s(from_str, to_str)
str.tr_s!(from_str, to_str)
str.unpack(format)
str.upcase
str.upcase!
str.upto(other_str) {|s| block }
```

A. Why does this class have so many methods?

B. Go through the methods listed and categorize them into five to ten major areas of responsibility.

C. Many of the methods have aliases (e.g., `next` and `succ`, `[]` and `slice`). What are the tradeoffs in having aliases?

D. Most String methods have two versions—for example, `str.reverse` and `str.reverse!`. (The first form returns a new string; the ! form changes the existing string in place.) What are the consequences of having the two types of methods?

E. On balance, do you consider the size of class String to be a smell?

F. In Java, class Object has 11 methods, whereas in Ruby and Smalltalk it has many times this number. Why the difference? Talk to a Java person and consider whether you think Ruby's version smells.

See page 218 for solution ideas.