Grades
- “Parameterized” tests in rspec-style frameworks
  - describes, its used to form data structure
  - each it must have unique composite name
- fromGrade
  - Elementary syntax of match/case: basePointsWithSwitch
  - guard capability: modifierPoints
  - Better to data-drive here: basePointsDataDriven
- toGrade
  - guard pattern can be more complex
  - nesting more logic inside case can be confusing

FizzBuzz
- standardIf: perhaps easier to write, 1-3 divs, 53% 3
- cachedIf: always 2 divs
- patternMatch: always 2 divs, tuple unapply, no guards

Types
- soundOf: match/case sensitive to type
- usually wrong if type is only discriminator
- use polymorphism instead: trait Soundable
- problems are caught at compile time not runtime

Recursive
- adding node values in binary tree
- simpler if parameter is option
- either way pattern match is clearer
- especially patternMatchSum: see how works later

PartialFunction
- total function: returns value for every input
- partial function: cannot produce value for some inputs
- function defs in Scala are syntactic sugar
- evaluate to instances of FunctionX
- partial functions override PartialFunction
- isDefinedAt and apply can be combined with
  - pattern-matching sugar

Unapply
- recalls; don’t hate me
- draw attention to test line 27
- no unapply: lots of logic

add unapply to companion object
    must accept single parameter
    must return Option[(A, B, …)]
    means one unapply per type per object
    can call directly
when used in match, logic much more compact
case class gives free unapply
possible but not recommended: other unapplys
    single non-boolean attribute: Option[A]
    boolean attribute: Boolean
    too verbose for effect
    non-extractor unapplys confusing