object.unapply - The Magic Behind Scala's Pattern Matching

- ***unapply*** is tuple-ware (h/t ?name?)
  - `case class Extract( a: String, b: Int )`
  - `def unapply(e: Extract) = Some( Tuple2(e.a, e.b) )`
- Pattern Matching Flavours
  1. match case
     - `x match { case Extract( a, _ ) => a }`
     - Scala does type checking and casting for you.
  2. variable definition
     - `val Extract( a, _ ) = x`
     - Beware of the `scala.MatchError RuntimeException`
  3. case as partial function
     - `list collect { case Extract( a, _ ) => a }`
     - `MatchError` cannot occur
  4. for ~~*comprehension*~~ expression
     - `for ( Extract( a , _ ) <- ... ) yield a`
- I think of `unapply` as turning an object inside-out. With `apply`, one creates an object from parts, whereas with unapply you get the parts of which the object was made.
- Matching is a recursive process of:
  1. `unapply`
  2. Is it `equals()`
  3. Rinse / repeat (with type checking and variable binding along the way).