```
class User(id: UserID, name: String,
  followers: Set[UserID] with LatencyBound(20 ms),
  timeline: List[TweetID] with LatencyBound(20 ms))

class Tweet(id: TweetID, user: UserID, text: String,
  retweets: Set[UserID] with Size(ErrorTolerance(5%)))

def viewTimeline(user: User) = {
  // `range` returns `Rushed[List[TweetID]]`
  user.timeline.range(0,10) match { // use match to unpack
    case Consistent(tweets) =>
      for (tweetID <- tweets)
        displayTweet(tweetID)
    case Inconsistent(tweets) =>
      // tweets may not have fully propagated yet
      for (tweetID <- tweets)
        // guard load and retry if there's an error
        Try { displayTweet(tweetID) } retryOnError
  }
}

def displayTweet(id: TweetID, user: User) = {
  val rct: Interval[Int] = tweets(id).retweets.size()
  if (rct > 1000) // abbreviate large counts (e.g. "2k")
    display("${rct.min/1000}k retweets")
  else if (rct.min == rct.max) // count is precise!
    display("Exactly ${rct.min} retweets")
  //...
  // here, `contains` returns `Consistent[Boolean]`
  // so it is automatically coerced to a Boolean
  if (tweets(id).retweets.contains(user))
    disable_retweet_button()
}
```