class Bookings {
    // array of dates containing names of person who reserved the listing (as an option type)
    // initialize the array to have None at each date
    private val reservations: Array[Option[String]] = Array.fill(30)(None)

    // compute how many dates the listing is NOT booked – for loops version
    def daysAvail(): Int = {
        var avail = 0
        for (guest <- reservations) guest match {
            case None => avail = avail + 1
            case Some(s) => Unit
        }
        avail
    }

    // compute available days with operations not for loops
    def daysAvail2(): Int = reservations.length - reservations.flatten.length

    // print the reservations with a for-loop
    def show() =
        for (i <- 0 to reservations.length - 1)
            print((i + ":" + reservations(i) + "", "))

    // print the reservations with a function operator
    def show2() = reservations.foreach(date => print(date + ":" + date + "", "))

    // the exception is similar to Java
    class NotAvailable(val date: Int) extends Exception {}

    // the method to check availability. Notice no throws annotation in Scala
    def confirmAvailableExn(startDate: Int, numNights: Int): Unit = {
        for (date <- startDate to startDate + numNights - 1)
            reservations(date) match {
                case Some(str) => throw new NotAvailable(date)
                case _ => Unit
            }
    }

    // Make a reservation – shows what a Scala catch block looks like
    def reserveCheck(startDate: Int, numNights: Int, forName: String) = {
        try {
            confirmAvailableExn(startDate, numNights)
            for (date <- startDate to startDate + numNights - 1)
                reservations(date) = Some(forName)
        } catch {
            case e: NotAvailable => println("Already booked on " + e.date)
        }
    }

    // Alternative confirmAvailable that returns a Boolean and uses list/array operations instead
    def confirmAvailable(startDate: Int, numNights: Int): Boolean = {
        reservations.slice(startDate, startDate + numNights).forall{
            elt => elt match {
                case Some(str) => false
                case None => true
            }
        }
    }
}