object Assignment\_10 extends App {  
  
 println("Task\_1:")  
 def pair: Iterator[(Int, Int)] = for {  
 a <- Iterator.from(1)  
 b <- 1 until a + 1 if a % b == 0  
 } yield (a, b)  
  
 println("Method next:")  
 var i = 0  
 var p = pair.buffered  
 while (i < 20) {  
 i += 1  
 println(p.next)  
 }  
  
 println("Method take:")  
 pair.take(20).foreach(println)  
  
 println("Task\_2:")  
 val Task\_2 = for {  
 a <- Yes("Hello")  
 } yield a  
 println(Task\_2)  
  
}  
  
class Maybe[A](var value: A) {  
 def map[B](f: A => B): Maybe[B] = {  
 val newValue = f(value)  
 new Maybe(newValue)  
 }  
 def flatMap[B](f: A => Maybe[B]): Maybe[B] = {  
 val newValue = f(value)  
 newValue  
 }  
 override def toString = value.toString  
}  
class Yes[A](value: A) extends Maybe[A](value) {  
 private var v: A = value  
 def getContent: A = v  
}  
object Maybe {  
 def apply[A](value: A): Maybe[A] = new Maybe(value)  
}  
object Yes {  
 def apply[A](value: A): Yes[A] = new Yes(value)  
}