```
(*  Brij (TM)
   Copyright (c) M. P. Trivedi 2016-2025.  All rights reserved.

   UndoModel

Override thus (see <a href='https://learn.microsoft.com/en-us/dotnet/fsharp/language-
reference/members/methods'>this</a>
   type Circle(radius: float) =
      inherit Ellipse(radius, radius, 0.0)
      // Circles are invariant to rotation, so do nothing.
      override this.Rotate(_) = ()

output:
all: ["testing here"; "testing here and there"; "testing here and there and where";
 "testing here and everywhere"; "done testing there"]
curr: "done testing there"
ptr: 0
undoBtn: "enabled" redoBtn: "disabled"
 ----------------------------
cmd: undo
all: ["testing here"; "testing here and there"; "testing here and there and where";
 "testing here and everywhere"; "done testing there"]
curr: "testing here and everywhere"
ptr: -1
undoBtn: "enabled" redoBtn: "enabled"
 ----------------------------
cmd: undo
all: ["testing here"; "testing here and there"; "testing here and there and where";
 "testing here and everywhere"; "done testing there"]
curr: "testing here and there and where"
ptr: -2
undoBtn: "enabled" redoBtn: "enabled"
 ----------------------------
cmd: redo
intptr: -1
all: ["testing here"; "testing here and there"; "testing here and there and where";
 "testing here and everywhere"; "done testing there"]
curr: "testing here and everywhere"
ptr: -1
undoBtn: "enabled" redoBtn: "enabled"
 ----------------------------
cmd: redo
intptr: 0
all: ["testing here"; "testing here and there"; "testing here and there and where";
 "testing here and everywhere"; "done testing there"]
curr: "done testing there"
ptr: 0
undoBtn: "enabled" redoBtn: "disabled"
 ----------------------------
eom

*)


namespace Trivedi

#nowarn "20" "25" "58" "66" "67" "64" "760" "1182" "1558"

type UndoMachine (li:string list, ptr:int) as m =
   let mutable currSt = List.last li
   let mutable st = li
   let mutable intPtr = 0
   let mutable undoBtn = true
   let mutable redoBtn = false
   abstract member add: string -> unit
   default m.add (s: string) =
      match intPtr with
      | 0 -> st <- st @ [s]
      | _ -> st <- st.[0 .. (List.length li) + intPtr]
      currSt <- List.last st
      m.resetBtns()
      m.stat()
   abstract member undo: unit -> unit
   default m.undo() =
      printfn "cmd: undo"
      m.btnStat()
      intPtr <- (intPtr - 1)
      currSt <- li.[(List.length st) - 1 + intPtr]
      m.resetBtns()
      m.stat()
   abstract member redo: unit -> unit
   default m.redo() =
      printfn "cmd: redo"
      m.btnStat()
      intPtr <- (intPtr + 1)
      printfn "intptr: %A" intPtr
      currSt <- if intPtr = 0 then List.last st else (li.[(List.length st) - 1 + intPtr])
      m.resetBtns()
      m.stat()
   member m.resetBtns() =
      let ll = List.length st
      let atTop = (ll = (-intPtr))
      match (ll, atTop) with
      | (0, true) -> undoBtn <- false
      | _ -> undoBtn <- true
      match intPtr with
      | 0 -> redoBtn <- false
      | _ -> redoBtn <- true
   member m.btnStat() =
      let u = if undoBtn then "enabled" else "disabled"
      let r = if redoBtn then "enabled" else "disabled"
      printfn "undoBtn: %A redoBtn: %A" u r
   member m.stat() =
      printfn "all: %A\ncurr: %A\nptr: %A" st currSt intPtr
      printfn " ---------------------------- "

[<AutoOpen>]
module FrmDef_Actual =
   open System

   let runTest() =
      let mach = UndoMachine(["testing here";"testing here and there";
      "testing here and there and where";"testing here and everywhere";], 3)
      mach.add("done testing there")
      mach.undo()
      mach.undo()
      mach.redo()
      [0..5] |> List.map (fun x -> mach.undo()) |> ignore
      [0..5] |> List.map (fun x -> mach.redo()) |> ignore

   runTest()
   printfn "eom"
```